

**EMPIRICAL EVALUATION OF DESIGN PRINCIPLES FOR
INCREASING REVIEWABILITY OF FORMAL REQUIREMENTS
SPECIFICATIONS THROUGH VISUALIZATION**

by

NICOLAS DULAC

B.Eng. McGill University, 2001

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

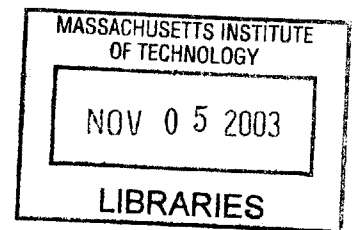
MASTER OF SCIENCE IN
AERONAUTICS AND ASTRONAUTICS

at the


MASSACHUSETTS INSTITUTE OF TECHNOLOGY
August, 2003

© Nicolas Dulac, 2003
All rights reserved.

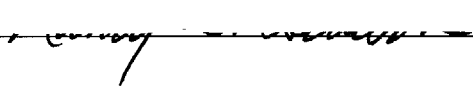
The author hereby grants to MIT permission to reproduce and
distribute copies of this thesis document in whole or in part.



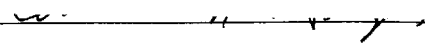
Signature of Author


Department of Aeronautics and Astronautics
August 2003

Certified by


Professor Nancy G. Leveson
Department of Aeronautics and Astronautics
Thesis Supervisor

Accepted by


Professor Edward Greitzer
H.N. Slater Professor of Aeronautics and Astronautics
Chair, Committee on Graduate Students

AERO

EMPIRICAL EVALUATION OF DESIGN PRINCIPLES FOR INCREASING REVIEWABILITY OF FORMAL REQUIREMENTS SPECIFICATIONS THROUGH VISUALIZATION

by

Nicolas Dulac

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the
Degree of Master of Science.

Abstract

As software systems become more pervasive in the aerospace industry, new techniques need to be developed that allow engineers to accurately review and understand the complex requirements specifications of these software systems. Several visualizations that provide a different view of formal specifications are proposed based on the experience of trying to manage the complexity of the MD-11 flight management system. A taxonomy for discussing these visualizations and a set of general principles that guide the development of visualizations for formal specifications are developed.

An interactive tool is developed to implement the visualizations and a user experiment is conducted using the tool to evaluate several of the principles. Subjects answer questions about the formal model of the annunciation process of the MD-11 vertical guidance system using both SpecTRM-RL and the visualization tool. The results of the experiment demonstrate the usefulness of formal methods in complex aerospace software systems and the potential of information visualization to increase the reviewability of formal specifications.

Thesis Supervisor: Dr. Nancy G. Leveson
Professor of Aeronautics and Astronautics

Acknowledgements

First and foremost, I would like to thank Professor Nancy Leveson, my academic advisor and thesis supervisor. Your enthusiasm, support and guidance during these two years at MIT have made this work possible.

I would also like to thank Professors Peggy Storey and Kim Vicente for your invaluable research advices and for your willingness to share your time and experience. My gratitude also goes to Professor Rakheja, for sharing your passion for research and for encouraging me to apply to MIT.

Many thanks go to Thomas Viguier, my French friend and colleague. Your knowledge and intelligence made of you an outstanding research partner.

Working in such a stimulating environment was an extraordinary experience. Many thanks go to my labmates for being so smart, yet so grounded and fun. Thanks Ed, Natasha, Mirna, Karen, Polly, Victor, JK, Elwin, Steph, Martin, Stella, Anna, and all those I forgot.

Thanks to all the volunteers who agreed to participate in the experiment. Your patience and insight were greatly appreciated. Thanks also to my two UROPs, Craig and Eric, whose work has helped me to complete this thesis on time, or at least not too late...

Also thanks to all my teammates on the MIT Varsity Hockey Team for providing much needed humor-based (read: Canadian jokes) stress relief during the past two years. Dump and bang, guys! Thanks to my coaches: Marky, Jimmy, Donnie and Froggy, for being such an example of dedication and for allowing me to extend my hockey career.

Katie... Thank you for everything. Thank you for being there for me during these years, and mostly, thank you for being such a loving, caring, and supportive person.

Many thanks go to my family, whose unconditional love and support over the course of my (long) studies have been essential for the completion of this thesis and every other thing I accomplished. Thanks Mom, Dad, Frank and Marie.

Table of Contents

Abstract	2
Acknowledgements	3
Table of Contents	4
List of Figures	7
List of Full-Page Figures	9
List of Tables	10
Chapter 1. Introduction	11
1.1 Formal Requirements Specification	12
1.2 Specification Reviewability	14
1.3 Measuring Reviewability	15
1.4 Factors Affecting Reviewability	16
1.5 Visualizing Formal Specifications	18
1.6 Visualization Design	19
Chapter 2. Background	20
2.1 State-Based Specifications	20
2.2 Formal Specifications Readability	22
2.3 Related Work on Visualization	23
2.3.1 Cognitive Aspects of Visualization	23
2.3.2 Visual Programming Languages	24
2.3.3 Large Media Visualization and Exploration	25
2.3.4 Human-Machine Interactions	27
2.4 SpecTRM	27
2.4.1 Intent specification	28
2.4.2 SpecTRM-RL	30
2.4.3 SpecTRM Interface	31
Chapter 3. Visualizations and Taxonomy	33
3.1 Sample Visualizations	33
3.1.1 Structural overview - (V1)	33
3.1.2 Question-Based Decision-Tree – (V2)	38
3.1.3 State Transition Diagram and Inversion – (V3)	43
3.2 Taxonomy of Visualizations	46
3.2.1 Scope	46
3.2.2 Content	46
3.2.3 Selection Strategy	47
3.2.4 Annotation Support	47
3.2.5 Support for Alternative Search Strategies	48
3.2.6 Static / Dynamic	48

Chapter 4. Requirements Specifications Visualization Design Principles	49
4.1 Minimize Semantic Distance	49
4.2 Match the Task being Performed	51
4.3 Support the Most Difficult Mental Tasks	51
4.4 Highlight Hidden Dependencies and Provide Context when Needed	52
4.5 Support Top-Down Review	53
4.6 Support Alternative Problem-Solving Strategies	54
4.7 Show Roles Being Played	55
4.8 Provide Redundant Encoding	55
4.9 Show Side Effects of Changes	56
Chapter 5. Experiment Design	58
5.1 Experiment Objective	58
5.2 Experiment Hypothesis	60
5.3 MD-11 FMS Formal Specification	60
5.4 Tools Description	64
5.4.1 SpecTRM	64
5.4.2 Visualization Tool	66
5.5 Experiment Methodology	66
5.5.1 Subject Selection	67
5.5.2 Tutorial	67
5.5.3 Experiment Questions and Tasks	68
5.5.4 Post-Experiment Analysis	69
5.5.5 Experimental Setup	71
Chapter 6. Experiment Results	72
6.1 Grading System	72
6.2 General Results	73
6.3 Performance Metrics Results	76
6.3.1 Answer Accuracy Results	76
6.3.2 Answering Time Results	80
6.3.3 Question Difficulty Results	82
6.4 User Preferences	84
6.5 Principle-Specific Discussion	85
6.5.1 Highlight Hidden Dependencies	85
6.5.2 Support Top-Down Review	87
6.5.3 Support Alternative Strategies	89
6.5.4 Provide Redundant Encoding	91
6.6 Experiment Limitations	92
6.7 Recommendations	94

Chapter 7. Conclusion	96
7.1 Attractiveness and Performance of Visual Representations	96
7.2 Validity of <i>Highlight Hidden Dependencies</i> Principle	97
7.3 Validity of <i>Top-Down Review</i> Principle	98
7.4 Validity of <i>Support Alternative Strategies</i> Principle	98
7.5 Validity of <i>Provide Redundant Encoding</i> Principle	99
7.6 Other Results and Observations	99
References	101
Appendix A. Full-Page Figures	104
Appendix B. Experiment Questions	113
Appendix C. Complete Quantitative Results	121

List of Figures

Figure 2-1. Simple state-machine of a car cruise control system	21
Figure 2-2. Intent specifications dimensions	29
Figure 2-3. SpecTRM visual overview of an altitude switch system	31
Figure 3-1. Legend key of the visual elements of the structural overview (V1)	35
Figure 3-2. A sample state-machine transition diagram and inverse transition diagram taken from the MD-11 vertical guidance specification. This state variable describes the vertical attitude of the aircraft during cruise	44
Figure 4-1. Sample graphical overview of the modes and state variables of a digital avionics system	54
Figure 5-1. Inside view of the MD-11 cockpit. The VG Annunciation Process sends feedback information to the highlighted display units	61
Figure 5-2. Detailed view of the main display units of the MD-11 cockpit	62
Figure 5-3. Detailed view of the Flight Control Panel (FCP) located on the glareshield panel of the MD-11 cockpit	62
Figure 5-4. Hierarchical context of the Vertical Guidance Annunciation Process function along with its main I/O devices	63
Figure 6-1. Summary of the average performance (answer accuracy) of each subject graded on a 0-10 performance scale	74
Figure 6-2. Summary of the average difficulty rating of each subject evaluated on a 0(easy)-10(difficult) scale	75
Figure 6-3. Summary of the average time spent answering each question	76
Figure 6-4. Average overall performance for subjects using the three different tool configurations	77
Figure 6-5. Average results obtained for each question part using the three possible tool configurations	78

Figure 6-6. Average time spent on each question part using the three possible tool configurations	81
Figure 6-7. Average overall difficulty rating for subjects using the three different tool configurations	82
Figure 6-8. Average difficulty rating for each question part using the three possible tool configurations	83

8\

List of Full-Page Figures (in Appendix A)

Figure A. Screen capture of the SpectRM GUI	105
Figure B. An organized overview of the view of the structure of the specification (V1)	106
Figure C. Sliced structural overview. The state variable <i>Origin of Level T_D</i> and its structural dependencies are emphasized over the rest of the model, which is preserved as context	107
Figure D. A Questions-based Decision Tree taken from the MD-11 Vertical Guidance Specification. The state variable <i>Active Add Drag Scenario</i> indicates which of the five possible scenarios for extending airbrakes is active, if any	108
Figure E. A Questions-based Decision Tree from the MD-11 Specification. The state variable <i>Origin of Econ Airspeed Target</i> determines how the Vertical Guidance System will compute the <i>Econ Airspeed Target</i>	109
Figure F. Effects of a slicing scenario applied on the state variable <i>Origin of Econ Airspeed Target</i> (Figure E). The states reachable under this scenario are highlighted. The active scenario specifies that the flightphase is ‘Descent’ and that the current operational procedure is ‘Descent Path’ or ‘Late Descent’	110
Figure G. Five of the six AND/OR tables necessary to describe the state variable <i>Active Add Drag Scenario</i> in the specification of the MD-11 Vertical Guidance Annunciation Process. This state variable specifies which of the five possible scenarios for extending airbrakes is active, if any	111
Figure H. Screen capture of the entire visualization tool	112

List of Tables

Table 5-1. Tool/Question combination for the twelve subjects 68

Table 5-2. Variables used in the experiment 69

Chapter 1

Introduction

In the last decade, the aerospace industry increasingly relied on software to replace functions traditionally performed by hardware devices. The increased prevalence of software systems has been accompanied by a dramatic increase in system size and complexity. This trend toward ever larger, complex, and integrated software systems inevitably increases the likelihood of subtle, undetected errors. This is a most important concern because software is now used to perform many safety-critical functions in aerospace systems, such as aircraft guidance and spacecraft attitude control. Examples of system failures caused by software errors abound: the 1999 Mars Climate Orbiter and Ariane 5 failures are just two recent examples. This threat will only increase in the next decade as software becomes even more pervasive. Leveson summarizes the current situation in Safeware [20]:

“Few systems today are built without computers to provide control functions, to support design, and sometimes to do both. Computers now control most safety-critical devices, and they often replace traditional hardware safety interlocks and protection systems—sometimes with devastating results.”

The growing size and complexity of software is a result of strong incentives to develop integrated systems providing more functionality at lower cost. Theoretically, the exponential increase in computing power available paves the way for such an increase in complexity and functionality. The common industry belief that software is infinitely flexible contributes to this increase in complexity by replacing well-proven traditional hardware systems with complex software systems including ever more functions. For example, the Boeing 777 includes more than four million lines of code distributed over 79 interdependent systems. This represents a six-fold increase over Boeing’s previous aircraft program.

This exponential growth in software size and complexity is not matched by a parallel improvement of software development methods. Most of the time, standard software development processes are used to develop safety-critical software systems without regard for the unique requirements of safety-critical system.

Contrary to popular belief, many of the most serious software errors result from flawed requirements specification activities rather than from coding mistakes. Most of the errors in software systems can be traced back to the requirements. Errors can occur if the requirements are wrong, incomplete, or misinterpreted. Requirements errors are not only difficult to detect, but they are the most expensive to correct at a later development stage because they were made early in the development process and most likely influenced a variety of design decisions. Exhaustive testing of large specifications is impossible because of the massive discrete state-space involved. Theoretically, formal methods and mathematical analysis present a solution to our inability to test even a small portion of the enormous state-space involved in today's complex digital systems. However, while automated analysis tools can find some types of errors, detecting many of the most serious semantic errors (e.g., will the software advance the throttle under unsafe conditions or will the software behavior lead to human errors in controlling the aircraft) requires human expertise. Validating software behavioral requirements in complex systems is a necessarily multidisciplinary problem involving a large number of engineering disciplines.

1.1 Formal Requirements Specification

Requirements specification is one of the earliest and most decisive stages of system development. The impact of the decisions – or mistakes – made during this phase will resonate throughout the whole system lifecycle. Some people argue that the requirements specification phase is not as critical for software systems because software is flexible and can be easily changed or updated at a later date. However, the apparent flexibility of software makes it a perfect candidate for late fixes when deficiencies are found in other parts of the system. As Shore [31] puts it: “*Software is the resting place of afterthoughts*”.

The requirements specification process includes many activities that often overlap. Among these activities are the specification creation, modification, validation, and review. A large number of people with various backgrounds participate in the specification process,

including systems engineers, developers, designers, QA people, HMI specialists, regulation experts, operators, customers, etc.

Irrespective of the process followed in order to obtain the specification, two types of requirements specification can be used. *Informal specifications* use plain English and ad-hoc diagrams to describe the system requirements. The organization and format of informal specifications is often dictated by the tool used to produce and manage it and by company policies. *Formal specifications* use a language with rigorously defined syntax and semantics to describe the requirements. The format of formal specifications is constrained by the properties of the language used. Z and Statecharts are two well-known formal specifications languages. Formal specifications are part of a larger methodology called formal methods. Clarke and Wing [5] define formal methods as “mathematically based languages, techniques, and tools for verifying and specifying hardware and software systems”.

Through the use of a formal language, formal specifications can promote a common understanding of the required functionality of the system. Since the language used to create the specification is rigorously defined, the specification will not be subject to misinterpretations resulting from users with various backgrounds interpreting ambiguous natural language expressions differently. The use of unambiguous language is a key for domain experts to validate that a specification describes a safe and useful system. All specifications are meant to be a clear means of communication between all the stakeholders involved in systems development. The advantage of formal specifications is that they are accurate and explicit, provided they can be read and understood by these stakeholders.

During a formal specifications process, a specification language is used to create a formal model of the system requirements. The mathematical foundation of formal specification languages allows for analysis of some properties of the specification. Among others, formal models can be analyzed to prove that they are deterministic and robust [14,23]. Because of the large amount of computation required to prove such properties for complex systems, these analyses are usually done through the use of automated tools. Another advantage derived from the use of formal specifications is that the formal models obtained can be executed in order to “see” the system in operation before it is implemented and make sure the resulting system will perform as expected. This is only possible through a rigorous specification of the externally visible behavior of the system. The benefits derived from the use of formal specifications can

result in safer and more reliable products, increased product quality, and reductions in cost and time-to-market.

However, formal methods are not widely used the industry. Clarke and Wing [5], and Gerhart and Craigen [9] have conducted surveys of the state of the practice in industry and academia. Except for a few successful projects in industry, formal methods remain limited to the academic world. Two main reasons may be advanced for this lack of industrial acceptance: most formal specifications are not easily readable, and they do not scale well to complex systems. Readability is arguably one of the most important properties of any specification [21]. Formal specifications will never be accepted if a large amount of training is required only to read them. Scalability is also a major concern. Most of the formal methods research work in academia use very small models. Proving properties of toy-size models may be an interesting intellectual exercise and can produce quality research work, but as long as the theory continues to be applied to unrealistically small models instead of complex real-world systems, the gap between academia and industry will not be bridged.

Making formal specification languages readable and scalable to large complex systems is a key factor in promoting the adoption of formal specifications by the industry. One of the objectives of the Software Engineering Laboratory (SERL) at MIT is to increase the readability and scalability of formal specifications to promote their industrial acceptance and profitability. Previous work by members of this laboratory has addressed the readability of the representations used to specify the externally visible behavior of complex systems [41]. This thesis concentrates on increasing the reviewability of formal specifications through the use of visualization.

1.2 Specification Reviewability

The review of specifications is a critical phase in systems development because undetected specification errors will ripple through the systems design and implementation phase. When detected at a later development stage, the cost of fixing these errors will many times higher than it would have been in the specification stage. The review of specifications typically includes two types of tasks. The first task is to ensure that the low-level systems requirements will successfully fulfill the high-level customer requirements. That is, making sure the system

specified will do what it was intended to do. Domain experts who did not necessarily write the specification usually do this type of review, often called specification validation. The second type of review task involves finding errors in the specification itself. These errors range from common typos to major inconsistencies. This type of review is often conducted by a variety of people with different backgrounds. For systems requiring certification, this often includes representatives from regulation agencies. This type of review can be compared to peer review in programming, and is often called formal review when the review sessions are conducted using a strictly defined format.

Some advocates of formal methods argue that mathematical analysis facilitates the review of formal specifications. Although it is true that many advantages are derived from mathematical analysis of formal models, it does not lessen the need for specifications to be readable. Experience has shown us that engineers will not put their confidence in a specification they cannot understand. Furthermore, the refinement of high-level customer requirements to low-level system requirements requires human expertise and cannot be fully automated. Previous work at SERL has also shown that automated analysis will not detect all types of errors, especially subtle errors requiring domain knowledge.

Additionally, reviewing specifications of complex systems is an inherently multidisciplinary activity, more so than any other systems development activity. In spite of their mathematical foundation, formal specifications need to be readable and understandable by any individual involved in the systems development process, including people with no knowledge of computer science or software engineering.

1.3 Measuring Reviewability

The preceding section explained why reviewability is such a desirable property of formal specifications. In order to evaluate the reviewability of formal specifications, reviewability metrics have to be defined. Reviewability can be defined as the efficiency of performing various reviewing tasks. This is also a very subjective definition and should be further refined into measurable components. Reviewing tasks will vary a lot according to the domain and function of the specified system. For example, a flight entertainment system onboard an airliner may be

highly complex, but will not be subject to the same type of thorough review as the landing gear monitoring system, a simple but critical system. The experience and knowledge of reviewers will also affect the efficiency of the review process; A difficult reviewing task for a novice may be trivial for a domain expert. As a result, objectively measuring the efficiency of review activities as a whole is more difficult than it appears at first. For this reason, review efficiency is easier to measure for given tasks, or category of tasks. It is believed that by combining the cumulative efficiency of a formal specification language to perform a set of typical tasks, a general assessment of the reviewability of a formal specification can be obtained.

For example, a typical reviewing activity involves searching for the information necessary to perform a task. A complex formal specification is a large, dense information space and locating the required information can be a daunting task. Although the use of computer tools facilitates the location of critical information over paper specifications, different navigation functions may be available to facilitate information searches. A very useful measure of the efficiency of an information search function is the *cost structure* of information [2]. The cost structure is defined as the reachability of information in a dense information space, in terms of the amount of effort necessary to locate the desired information. A good cost structure will reduce the information search time and will make the critical or most often needed information available first. Since all reviewing tasks include information searches in some ways, the cost structure of an information space is an easy to evaluate metric that can be used, among others, to assess the reviewability of formal specifications.

1.4 Factors Affecting Reviewability

In order to evaluating the reviewability of a specification language, it is important to identify the factors that will affect this property. As mentioned previously, the difficulty or complexity of the reviewing task to be performed, as well as the experience of reviewers are two primary factors that will affect the apparent efficiency of a formal specification review process. Other factors stand out as equally important, although they might not be exhaustive, neither completely independent from each other.

The effectiveness of the notation or representation used to present the information is an important factor. It is possible, through experimentation, to objectively evaluate the effectiveness of different representations. However, even if we can prove that a representation is, on average, better than another, the efficiency of a representation can only be evaluated in the context of a certain type of users performing a certain type of tasks. A good quality representation can perform well for most reviewing tasks while being very difficult to use for a task in particular. The personal experience and background of reviewers may affect their performance and preferences for different notations. Nevertheless, the representation used affects the specification reviewability. There is no such thing as a perfect representation, but it is possible to say that some representations are much better than others.

Regardless of the representation used, the availability of powerful features that modify the information available in the specification also greatly affects reviewability. Some of these operations can be as simple as a text search that quickly locates keywords in the specifications. More powerful features include the ability to simplify the specification based on an operating scenario, or the ability to analyze a system model for completeness. Such tasks that support the reviewing process but that are tedious to perform manually should be automated as much as possible. Powerful features can greatly ameliorate the efficiency of a specification reviewing process, but they should always be well understood, and used with caution since they cannot replace experience and good sense.

Specification complexity is another factor that affects reviewability. Software systems are usually more complex than mechanical or physical systems because they are not subject to the same physical constraints. While physical systems have to comply with nature's laws and production constraints, software is infinitely malleable "thought stuff" that can be reproduced at will. Also, when compared to computer hardware, the complexity of software systems tends to be more difficult to manage because software is more organic and cannot take advantage of the repetitive structure of computer hardware. Our experience in trying to create and understand very large specifications for systems such as flight management, collision avoidance, and air traffic control has shown that even with a formal notation designed with readability in mind, the complexity of the behavior being described overwhelms the reader. Not only is it difficult to provide notations that can be reviewed by people with different backgrounds and expertise, but

for complex systems, most users (even the authors of the specification) need help in managing complexity.

Size is also a critical factor. The discrete mode logic for an aircraft flight management system may require thousands of pages of formal logic to specify in adequate detail. The review of such specifications by domain experts or even by those who are expert in the formal notation itself is a daunting task. The size of a specification document affects the user's cognitive workload. Even locating the necessary information within a specification hundreds of pages long can be a tedious task. Trying to build up a mental model of the dependency structure of only a few elements in the specification usually overloads the user's short-term memory. Navigating through large specification document is difficult and overloads the user with detailed information that degrades his or her ability to see the big picture.

Specification size and complexity are factors that are often out of the reviewer's control. However, finding ways to manage them more efficiently can lessen their effect. Powerful tools and functions can help navigating and understanding large, complex formal specifications. Since we are trying to increase the human's natural ability to review complex systems specifications, we need to find ways to extend the human's cognitive limits by amplifying cognition. Humans use external tools to stretch their cognitive limits by minimizing the use of internal working memory. For example, humans usually cannot perform long divisions without the use of paper and pencil. We believe that properly designed information visualization tools can be used to stretch the cognitive limits of the reviewers of formal specifications.

1.5 Visualizing Formal Specifications

In a compilation of papers on information visualization [2], Card, Mackinlay and Schneiderman define information visualization as "the use of computer-supported, interactive, visual representation of abstract data to amplify cognition". This definition is slightly restrictive. For example, a hand-made static representation of the inputs and outputs of a system could also be considered as an information visualization medium. However, since modern formal specification tools are computer-supported and interactive, the original definition will be used.

Diagrams are a universal way to amplify cognition. Diagrams are often used in a non-systematic way to explain parts of informal specifications. However, the preciseness of formal specifications provides the advantage of permitting the automatic generation of diagrams from the formal model itself.

Visualization clearly has an effect on cognition, but it is difficult to explain and quantify. Many cognitive scientists and computer scientists have endeavored to explain the appeal of graphical programming languages but could not reach an agreement. Our task is to design visualizations of formal specifications that will improve the reviewing process by taking advantage of the cognitive benefits derived from the use of visual representations.

1.6 Visualization Design

Unfortunately, there are few principles to follow when designing interactive or even non-interactive graphical and symbolic notations for visualizations of formal software requirements specifications. Most of the research work was done in other fields and cannot be directly applied to solve the problems specific to formal specifications. Some of the related work suggest principles to be used for the design of effective visualizations in fields such as programming, Internet navigation and software re-engineering. We believe that some of these principles can be adapted to fit within the context of formal requirements specifications. The research work presented in this thesis includes the introduction of a set of principles to be used for the design of effective visualizations for formal requirements specifications. Based on these principles, interactive visualizations were designed and evaluated through user studies in order assess the validity of the proposed design principles.

The following chapter provides background work on the reviewability of formal specification and a survey of related visualization work. Chapter three presents sample visualizations and a taxonomy used to classify the visualizations created. Chapter four introduces nine design principles for designing visualizations of formal requirement specifications. Chapter five explains the design of a user experiment intended to evaluate the design principles. Chapter six presents and discusses the results obtained in the experiment, and chapter seven summarizes and concludes this thesis.

Chapter 2

Background

The research work presented in this thesis builds upon our own experience in creating and reviewing complex requirements specifications as well as research in other fields such as formal languages readability, intent specification, information visualization, and cognitive engineering. This chapter presents the background research upon which our research is built.

2.1 State-Based Specifications

Formal specifications use a formally defined model to make statements about the system behavior. We use state-machines as the basis for the specification of our formal model because 20 years of empirical work have shown that they are the most easily understood and adopted by engineers. However, formal models can be built using other methods such as the abstract model specification used in Z [32] or the algebraic specification used in Larch [12].

State-machine specifications explicitly describe system behavior by a set of states and define operations as transition between states. A state-machine specification includes states and their possible values as well as well-defined transition conditions on those values. Figure 2-1 shows a sample state-machine of a car cruise control model along with the conditions that govern the transition between states and the prescribed system behavior associated with each state. The cruise control state variable can be in one of the following four states: *Cruise Control Off*, *Cruise Control On - Standby Mode*, *Increasing Speed* and *Maintaining Speed*. The arrows specify transitions between states. Transition arrows are defined by source state and destination state, and are labeled with triggering conditions. A transition between two states is fired when its associated triggering conditions are met. For example, the cruise control transitions from the state *Cruise Control Off* to the state *Cruise Control On - Standby Mode* when the driver turns the cruise control on. During this transition, the cruise control is initialized, corresponding to the prescribed action specified along with the transition condition. The state variable Cruise Control is a single state variable made up of four possible states. State-based formal specifications for

real systems are made up of several state variables that depend on one another. A drawback of state-machine specifications is the explosion of states that occurs when specifying complex systems.

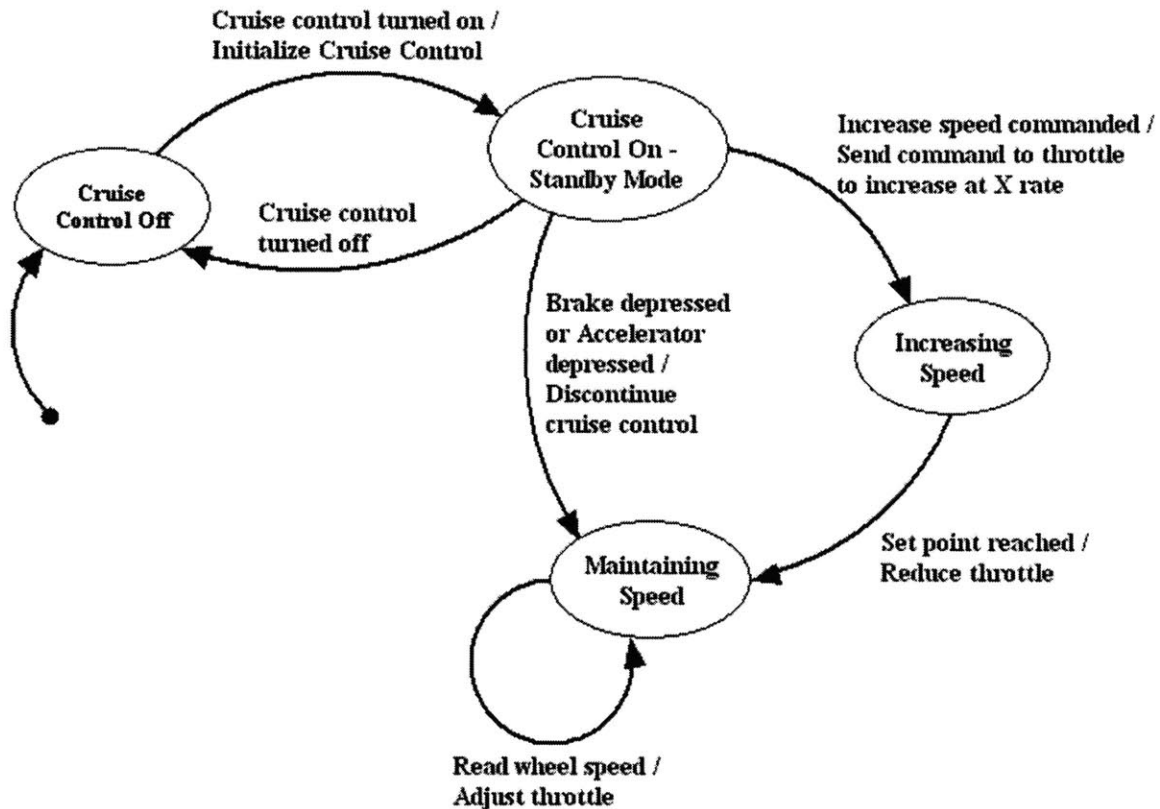


Figure 2-1: Simple state-machine of a car cruise control system.

State-machines are abstract entities, and as such, they can be specified using many different notations, both visual and textual. Figure A shows a visual representation of the state-machine that is very similar to the visual representation used in Statecharts [13]. Such representations work well for small models but become problematic for state machines including either a large number of possible states or complex transition conditions. All representations are not equal and some are better than others for specific tasks. The next section presents related work in evaluating the effectiveness of different representations.

2.2 Formal Specifications Readability

While formal methods have been successfully applied to hardware in industry, their use for software applications in industry have been very limited despite their large potential. Many reasons can be hypothesized to be the cause of this lack of widespread adoption of formal methods by the software industry. Since formal models are based on discrete mathematics and logic, their use requires some training in these fields of discrete mathematics. Engineers responsible for the specification of complex software systems typically do not possess the knowledge required to take full advantage of formal methods. Even if training in discrete mathematics was readily available for every engineer involved in the use of formal methods, the notations used in formal languages would discourage the use of such methods for complex projects. In fact, many formal specification languages often appear cryptic and require extensive effort to read and comprehend. Since the discrete mode logic for a flight management system requires hundred of pages of formal logic to specify, the use of an unreadable specification language prevents the use of formal methods altogether for systems larger than a simple toy or an elevator control system. Developing specification languages that are easily readable by every person involved in the specification and implementation of complex systems is the first building block necessary for the adoption of formal methods in the industry.

Formal specification tools and languages must take advantage of the mathematical tools and computing power available while hiding the mathematical representation of the system from the users. Just as it is possible for everyone to use a computer and navigate the internet without having the knowledge necessary to write a “hello world” program, engineers should be able to take advantage of formal methods without knowing about the underlying discrete mathematics upon which the methods are based. The formal methods field is currently going through this maturing process where the tools become powerful and simple enough so they can be used without advanced knowledge of the underlying mathematics. An example of another field that went through this maturing process is finite elements analysis, which went through a rapid maturing phase as the availability of computing power increased. Fifteen years ago, despite the well-established mathematical basis of finite elements and material science, developing a model of a simply loaded cantilever beam was a difficult task requiring much programming effort and scientific knowledge. Nowadays, a three-day seminar in finite elements analysis with a

commercially available package provide enough knowledge and skills for engineers to build complex models such as wing assemblies and to collect most of the benefits associated with the use of finite-elements analysis.

As mentioned previously, we use state-machine models because they have been widely used in a variety of large projects and are generally well accepted and understood by engineers. Starting from this hypothesis, since computing power is readily available, the first and arguably most important step is to provide a formal specification language that will be easily readable by an audience composed of people with different background and expertise including system designers and developers, customers, users, certifiers, etc. Zimmerman et al. [41] conducted an experiment in order to determine how various factors of state-based specification language design affect the readability of formal specifications using aerospace applications. Six factors were tested including the representation of the overall state-machine structure, the expression of triggering conditions, the use of macros, the use of internal broadcast events, the use of hierarchies and transition perspective. The results showed that the tabular representation of both the state-machine structure and the triggering conditions were better accepted and easier to use. As such, a specification language using tabular representations of state-machine structure and triggering conditions will be the basis against which our specification visualization tool will be compared.

2.3 Related Work on Visualization

There exists a large literature on the visualization of abstract quantitative data. Tufte's work [36] on the visual display of quantitative information, for example, is largely recognized and accepted as the reference for designing visualizations of quantitative data. However, very little work has addressed the visualization of processes or systems, which present very distinct difficulties, mostly because of the dynamic nature of the information to visualize. This section presents some work on the use of visualization in related fields that could potentially be applied to the visualization of formal specifications.

2.3.1 Cognitive Aspects of Visualization

Larkin and Simon [18] pioneered the research into the impact of visualizations on cognition. They first distinguished between *sentential representations*, whose contents are stored in a fixed

sequence like the propositions in a text, and *diagrammatic representations* whose contents are indexed by their position on a 2-D plane. While these two representations may contain the same information, their cognitive efficiency may be different. Certain features are more easily extracted from diagrams than from sentential representations. For example, adjacent triangles are easy to find visually, but require a potentially elaborate search through a sentential representation. Diagrams can also group together related concepts. Sentential representations may store related items in separate areas, thus requiring extensive search to link concepts.

Research has shown that the effectiveness of visualizations is highly dependent on the particular task that a user is trying to complete [3,26]. In other words, a given visualization may be ideal for presenting information for one task but may hinder the completion of a different task. There are many ways that representations can affect task performance. For example, visual representations can draw attention on certain aspects of the information that support problem solving. Casner and Larkin have suggested that good representations reduce the amount of cognitive processing in two ways: (1) they allow users to substitute quick perceptual inferences for more difficult logic inferences, (2) they reduce the search time for the information required to perform a task. On the flip side, representations can distract from the information that support problem solving by drawing attention to other information irrelevant to the task at hand. Good representations can shift the cognitive load, balancing the use of mental resources, shifting attention, and creating perceptual cues, but poor representations can create additional tasks or make the tasks more difficult to perform.

2.3.2 Visual Programming Languages

There has been a lot of research in visual programming languages and on the use of visualization to support program comprehension. Fitter and Green [8] proposed five principles of diagrammatic notation that may be applicable to formal specifications: (1) *Relevance* states that only relevant information should be encoded perceptually rather than symbolically, (2) *Restriction* states that notations should restrict the user to forms that are comprehensible, (3) *Redundant recoding* should be used for critical parts of the information, (4) Notations should *Reveal* the underlying processes that they represent, and should be used in the context of a *Responsive* interactive system, and (5) Notations should be readily *Revisable*. Redundant recoding, for example, is a very useful principle that is applied when specifying the same

information in two different ways, each of which simplifies different cognitive tasks. It can also be used to emphasize certain information. When a piece of information is especially important to a user's task, or if it is critical to the overall structure of the information, it is helpful to present a high-level view in a perceptual form, while at the same time presenting the detail in symbolic notation.

The most useful notations contain both symbolic and perceptual elements. In some cases they are independent and in other cases, they are logically redundant. An example of this is the use of indenting, color-coding, parenthesis matching and other cues to make programs more legible. These perceptual cues have been called *secondary notation* because they convey additional meaning above and beyond the "official semantics" of the language, or they disambiguate syntactic structure in order to assist in interpreting semantics. Petre [24] stresses that graphics do not guarantee clarity and that good graphics relies on this secondary notation, which is crucial for comprehensibility. Poor use of secondary notation is not merely neutral; it can confuse and mislead. It is argued that the appropriate use of secondary notation differentiates experts and novice visualization designers. Even though empirical studies did not find visual programming inherently superior to text based programs, visual programming is perceived as being more accessible, easier to understand, richer, and providing the "gestalt effect": providing an overview permitting the emergence of the overall structure and promoting the construction of a mental model. In itself, this positive image and appeal may have some important value.

Blackwell et al. [1] reviewed some empirical studies on the use of diagrammatic notations in programming and proposed a set of cognitive dimensions for visual language design, intended to be used as a vocabulary of terms describing cognitively-relevant aspects of structure of an information artifact. Example dimensions are closeness of mapping, consistency, role-expressiveness, and visibility. Because the context of visual programming languages is in some ways similar to that of specification languages, some of this work can be modified to fit our purpose.

2.3.3 Large Media Visualization and Exploration

The overwhelming increase in the size and complexity of information available from various sources has promoted some good research on the visualization of database data and high-

density information spaces such as the Internet. The sheer size of specification documents cannot be compared to the size of large databases. Nevertheless, specification documents of complex systems such as flight management systems of modern airliners span thousands of pages. Consequently, users need help in navigating through these documents and in finding the information they require to perform a task. Some of the work done in the visualization of large media such as databases, web searches, and complex programs can be applied to the navigation of complex formal specification documents. This section provides a sample of related work that could be useful for designing visualizations to help navigate and find information in large specifications.

Storey et al. developed a cognitive framework of design elements to be considered during the design of a software exploration tool [34]. Software exploration tools provide graphical representations of static software structures linked to textual views of the program source code and documentation. This framework contains two sets of factors to support the variety of comprehension strategies used by programmers during software exploration and to reduce cognitive overhead as they explore and try to construct a mental model of the software. The framework has been applied to the design and evaluation of a software exploration tool called ShriMP Views (Simple Hierarchical Multiple-Perspective) [33]. Although this framework has mostly been applied and evaluated using tools designed to visualize the structure and code of Java programs, the same combination of complex high-level structure and low-level details can be found in formal specifications, suggesting a possible adaptation of some parts of the framework to specification visualization.

Currently, navigating through large specification documents is in many ways similar to navigating through hypermedia documents using Internet browsers. A hypermedia document contains related and linked representations of an information space. Many of the difficulties experienced by users of specification documents are similar to those experienced by hyperdocument readers. Thuring et al. [35] define the readability of a document as the “mental effort spent on the construction of a mental model that represents the objects and semantic relations described in a text”. They state that assisting the users in the construction of their mental model can be done by strengthening factors that support this process and by weakening those that impede it. Two factors are identified as crucial in this respect: *coherence* as positive influence and *cognitive overhead* as negative influence on readability. A document is coherent if

a reader can construct a mental model from it that corresponds to facts and relations in a possible world [17]. The authors distinguish between *local* and *global* coherence and provide tips to increase document coherence. Conklin [6] defined cognitive overhead as the “additional effort and concentration necessary to maintain several tasks or trails at one time.” Cognitive overhead can be reduced by improving *orientation* and facilitating *navigation*. In an information space, orientation facilities are meant to help readers *find* their way and navigation facilities enable readers to actually *make* their way. Several design issues are brought up in order to reduce *cognitive overhead* and increase document *coherence*.

However, on certain specific points, searching for information in a specification document is different from surfing the web. For example, because of the strong coupling between different parts of the specification document, specification readers frequently need to navigate back and forth between elements. On the other hand, internet users navigate in a relatively linear way, from one document to another, while occasionally coming back to a previously viewed document (or node), usually to branch out to a new location. In this sense, the concept of unbreakable bi-directional links with strict version control proposed by Ted Nelson [25] in his ongoing Project Xanadu would be more helpful for specification navigation than for Internet. Also, information in formal specifications is often “modularized” into elements and even when dealing with complex specifications, it is always possible to extract the input-to-output mapping of an element within the system specification. We could see this as being able to observe the “position” of an element within the system boundary. This is obviously not possible while navigating the Internet, mostly because the lack of a well-defined boundary.

2.3.3 Human-Machine Interactions

Some research on human-computer interactions is also applicable to our problem. Designers of interfaces like those of requirements specification tools, need to select the appropriate level of abstraction, determine how to show relationships, provide context for the information, and build conceptual spaces using frames of reference [26].

2.4 SpecTRM

SpecTRM (Specifications Tools and Requirements Methodology) is a tool designed to provide bridges between diverse groups of system designers and implementers. It uses hierarchical

abstractions based on goals or purpose to deal with complexity and includes a formal modeling language (SpecTRM-RL) designed for readability. This formal foundation allows formal analysis and execution of the models built. This section describes SpecTRM and some of the related research work upon which it was built.

2.4.1 Intent specification

Intent Specification is an approach to writing system specifications based on research in systems theory, cognitive psychology, and human-machine interactions. Its goal is to provide specifications that support human problem solving and the tasks that humans have to perform in software development and evolution [22]. A second goal of intent specifications is to provide a better integration of formal and informal aspects of software development and enhance their interactions. While formal techniques are useful in some parts of the development process and are crucial in developing software for safety-critical systems, informal techniques will always make up a large part of complex software development efforts. To be useful to humans in solving problems, specification language and system design should be based on an understanding of the problem to be solved or the task to be performed. The language used to specify a problem has a direct effect on human problem-solving ability and affects the type of errors made while solving those problems. An approach to building human-centered specifications has to take into account what is known about human capabilities and limitations.

Cognitive psychology has established that the representation of a problem can affect the human problem-solving performance. In fact, the representations provided to the problem-solver either degrade or support performance [39]. Providing assistance for problem-solving, then, requires the development of a theoretical basis for evaluating whether a representation supports effective problem-solving strategies. Moreover, in addition to matching a representation to effective problem-solving strategies, the evaluation of a representation has to take into account the experience and background of specification user. In fact, it is likely that the different users of a specification will have different mental models of the structure and functioning of the system. Rasmussen has shown that problem-solving strategies are highly variable [29]. Problem-solving strategies do not only vary among individuals, but users often vary their strategy dynamically during problem solving, often as a reaction to difficulties encountered along the solution path. In the absence of a solid theoretical basis for evaluating and matching problem solving strategies to

tasks and users, specifications should support all possible strategies that may be needed for different users to perform a task. Since it is not possible to think of all the possible strategies that can be used to perform a task, the objective of specification language design should instead be to make it easy for users to extract and focus on the important information for the specific task at hand without assuming particular mental models or limiting the problem-solving strategies employed by the users of the specification [22]. Intent specification provides an approach to achieve this goal.

Intent specifications are organized along two dimensions: Intent abstraction and part-whole abstraction (Figure 2-2). The horizontal part-whole abstraction allows the users to change their focus of attention to more or less detailed views within a model or level. The vertical intent dimension has seven hierarchical levels, each level providing intent information (“why”) about the level just below while each level provides realization information (“how”) about the level above. Levels three and higher contain information that is most familiar to system engineers.

	Environment	Operator	System and Components	V&V
Level 0	Project management plans, status information, safety plans, etc.			
Level 1 System Purpose	Assumptions Constraints	Responsibilities Requirements I/F Requirements	System Goals, High-level Requirements, Design Constraints, Limitations	Hazard Analysis
Level 2 System Principles	External Interfaces	Task Analyses Task Allocation Controls, displays	Logic Principles, Control Laws, Functional Decomposition and Allocation	Validation Plans and Results
Level 3 Blackbox Models	Environment Models	Operator Task and HCI Models	Blackbox Functional Models, Interface Specs	Analysis Plans and Results
Level 4 Design Rep.		HCI Design	Software and Hardware Design Specs	Test Plans and Results
Level 5 Physical Rep.		GUI and Physical Controls Designs	Software Code, Hardware Assembly Instructions	Test Plans and Results
Level 6 Operations	Audit Procedures	Operator Manuals Maintenance Training Materials	Error Reports, Change Requests, etc.	Performance Monitoring and Audits

Figure 2-2: Intent specifications dimensions

The blackbox behavior specification included in level three specifies the system components and their interfaces, including the human components [22]. The behavioral specifications at this level are purely blackbox and describe the relationships between inputs and

outputs for every component in terms of externally visible variables, objects, and mathematical functions. The language used to specify each component can vary. SpecTRM-RL, a state-based formal specification language developed by Prof. Leveson and her students was used in this experiment.

2.4.2 SpecTRM-RL

SpecTRM-RL (Specification Tools and Requirements Methodology – Requirements Language) is a state-based formal specification language intended to assist engineers in managing the requirements, design, and evolution process. It shares many similarities with its predecessor, RSML [19] (Requirements State Machine Language), a state-machine based specification language based on Statecharts [13]. SpecTRM-RL has an underlying formal logic that allows a model of the system specification to be analyzed for consistency and completeness [23,14]. It also allows models to be executed providing a set of inputs in order to verify the blackbox behavior of the specified system and to find errors early in the development process. SpecTRM-RL emphasizes readability and requires little training. Readability is achieved mostly through the use of AND/OR “transition tables” to specify the conditions where state variables will switch from one state to another, which have been found to be easier to read and to understand by people with different backgrounds than representations such as predicate logic or logic gates [41]. SpecTRM and RSML have been successfully used in the specification of several large systems, such as TCAS II, a complex Traffic Collision Avoidance System.

Several different elements are used in a SpecTRM-RL formal model. State variables and their possible values are the basis for the state-machine model upon which SpecTRM-RL is built. They represent state that can be inferred from the input/output relationship of the system. Modes are high-level state values that represent the supervisory and control mode of the system. Macros are Boolean abstractions used to increase intellectual manageability [41] of the specification by reducing the size of the AND/OR transition tables. Devices are external to the system and are input sources and output sinks. Inputs and outputs are required for the system to communicate with the outside world and are the only information allowed to cross the system boundary. Because of the functional modularity of most complex engineering systems, inputs and outputs are grouped by corresponding sending/receiving device. Messages are packets of information sent or received by a device. Functions calculate output and intermediate values.

SpecTRM-RL also provides a visual overview of the formal model that includes four main parts: (1) A specification of the supervisory modes of the controller being modeled, (2) A specification of its control modes, (3) A model of the controlled process (plant in control systems theory) including the inferred operating modes and system state, and (4) A specification of the inputs and outputs to the system. The visual overview displays the current state of the system as it changes over time when the model is executed. Figure 2-3 provides a sample visual overview of a simple altitude switch formal model. This simple model uses input values from two digital altimeters and one analog altimeter to automatically lower/raise the landing gear of an aircraft when the aircraft altitude goes below/above a predetermined threshold.

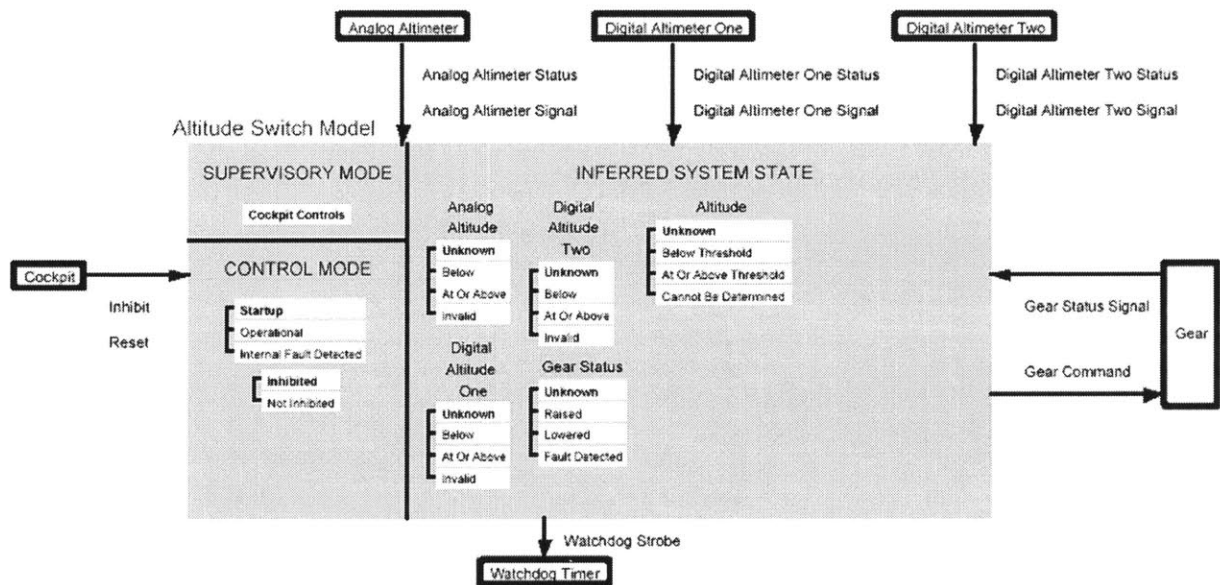


Figure 2-3: SpecTRM visual overview of an altitude switch system

2.4.3 SpecTRM Interface

Specifications of complex systems quickly become very large. Consequently, users of formal specifications need help in navigating through the document and in locating the required information. SpecTRM provides a project browser panel on the left that displays the structure of the document and helps users in finding information and in navigating through the multiple levels of the intent specification. All the information contained in the specification is displayed

in the formatted text panel on the right. Figure A provides a snapshot of the SpecTRM interface. SpecTRM includes Forward/Back navigation buttons that allow users to jump back and forth between different locations in the document just like it is possible to do in a web browser. SpecTRM provides automatic referencing in the Level 3 formal specification so that hyperlinks are automatically generated between linked elements. Hyperlinks can be manually generated in order to link different parts of the document and to provide tracing between levels. Tracing is extremely important in order to make sure that each high-level requirement has a corresponding lower-level implementation and that each implemented feature traces up to a high-level requirement so that no unwanted features are implemented. Hyperlinks greatly facilitate the navigation through the document but often produce disorientation and loss of continuity when used intensively in large documents. SpecTRM also provides a text search tool that comes in handy when the user does not know beforehand the location of some desired information.

Chapter 3

Visualizations and Taxonomy

While creating a SpecTRM-RL model based on an existing textual specification of the Vertical Guidance system of the MD-11 airliner, it quickly appeared that some form of visualization was needed to understand the functioning of such a complex system. Many different graphical representations were created manually in order to answer specific questions about the specification. A few of those graphical representations appeared to be very useful to understand the structure and functioning of formal specifications. However, the amount of efforts and time required to sketch those representations was a major shortcoming. It was decided to create a visualization tool that would automatically generate the most promising graphical representations based solely on the information included in the specification. The first section of this chapter introduces three visualizations that were used as the basis for the creation of our automated visualization tool.

It quickly appeared that we were lacking an appropriate vocabulary to discuss and classify the visual representations we created. In order to overcome this problem, we created a taxonomy of visualizations for formal specifications including six different dimensions. The last section of this chapter presents this taxonomy and briefly explains how it can be used to classify the visualizations created.

3.1 Sample Visualizations

This section describes three visualizations that were initially created to support the specification of the MD-11 FMS, and later included in an automated formal specification visualization tool.

3.1.1 Structural overview - (V1)

Figure B presents an interactive graphical overview (referred to as V1) of the dependency structure of the elements contained in the formal specification. The structural overview visually displays the elements contained in the model, along with the dependency relationships between the model elements and the devices external to the system.

Motivation

A quality mental model of the specification dependency structure provides many benefits for the users. For example, it allows the users to easily see how changes in one part of the specification will affect the overall system. It also makes it easier for the users to quickly identify errors such as unintended or missing dependencies between elements and to remove unnecessary coupling between elements or parts of the specification. The structural overview visualization was created to compensate for our inability to create a mental model of even small parts of the MD-11 FMS specification.

SpecTRM provides all the information necessary for the users to create a mental model of the dependency structure of the model. However, creating a high quality mental model of the specification structure requires such a large amount of cognitive effort to remember element dependencies that it is practically impossible. The structural overview makes the dependency structure of the specification explicitly visible, avoiding the need for the users to keep in memory the dependency relationships between the many elements. It is used as a basis for navigation and provides access to lower-level detail information about the behavior of each element of the model.

Description

The structural overview is an organized directional graph representation of the dependency structure of the specification. Figure B shows the structural overview of the annunciation process of the MD-11 Vertical Guidance system. It is made of nodes (boxes) representing elements of the models and arrow-links representing the dependencies between those elements. The arrows represent dependency relationships between elements. An arrow pointing from A to B, for example, means that the value of element B depends directly on the value of element A. The structural overview uses perceptual cues such as color, shape and position to encode critical information about the model elements and their interdependencies. Different colors are associated with different elements and dependency types. Figure 3-1 provides a legend key of the visual elements contained in the structural overview. The position of the elements in the model is arranged to provide a natural top-to-bottom input-to-output flow of information. As

such, the input devices are positioned on top of the model, the output devices are positioned at the bottom, and input-output devices are positioned on the side. Users can directly manipulate the elements on the screen. They are provided with intuitive, easy-to-use functions to select, move, resize or group elements. The structural overview includes only the information necessary to visualize the dependency structure of the model but provides access to detailed information when needed.

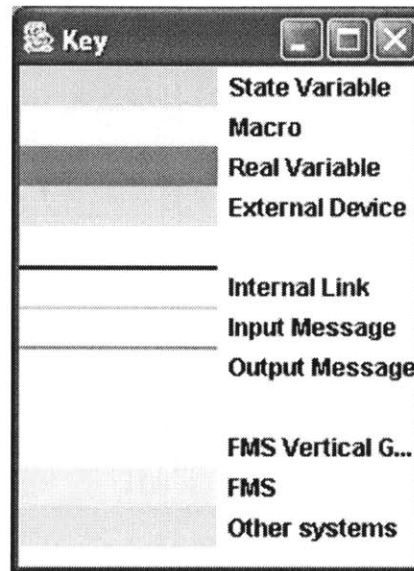


Figure 3-1: Legend key of the visual elements of the structural overview (V1)

The structural overview also provides context information about the system boundary and the hierarchical “position” of the system within its global operating environment. For example, in figure B, the elements that belong to the specified system, in this case the *Vertical Guidance Annunciation Process* system are positioned inside the white rectangle with a thick black boundary.

The *Vertical Guidance Annunciation Process*, along with the *VG Interpretation Process* and the *VG Guidance Process* are three functions of the *Vertical Guidance System* of the FMS. They are positioned inside the light gray rectangle, which represents the boundary of the Vertical Guidance system.

The Vertical Guidance system itself is a sub-system of the Flight Management System (FMS) and is positioned within the boundary of the FMS, along with five other sub-systems.

The devices external to the FMS, such as the ADC (Air Data Computer) and the FCC (Flight Control Computer) are positioned outside the FMS boundary. When multiple coupled sub-systems are specified, the structural overview allows the users to easily choose which sub-system specification will be visualized.

It can be seen in figure B that modern engineering systems such as flight management systems typically exhibit a high level of coupling between elements within the system and with devices external to the system. High coupling creates occlusion problems when trying to visualize the structure of specifications. Occlusion problems are inherent to any attempt to visualize high-density information spaces and can distort a user's perception of the information space or make it difficult to perceive patterns. Much work has addressed the problem of visualizing occluded high-density information spaces while maintaining situational context [2]. The structural overview deals with complexity and occlusion by allowing direct manipulation of the objects on the screen and simplification of the model structure based on techniques such as *slicing* and *filtering*. Simplifications through *slicing* are based on the dependency relationships between elements and simplifications through *filtering* are based on common attributes of elements. The interactive slicing tool provided with the structural overview allows the users to perform multiple levels up and/or down slicing of the specification based on a selected element. Figure C shows the result of using a complete input-to-output slicing on the specification based on the element *Origin of Level T_D*. When using simplification strategies like the interactive slicing tool, the information selected to be on a *slice* is put in focus through the use of larger fonts and thick borders, while the unselected information is grayed out and sent to the background as *context* information. At all times, the relative position of each element is conserved in order to ensure spatial continuity and avoid disorientation. Interactive slicing is available to users through a simple double-click on the chosen element. Pull-down menus are used to adjust simplification parameters.

Expected Advantages

It is believed that explicitly mapping the structure of a specification to a 2- Dimensional visual space will provide many benefits for the users:

- 1- It removes the need for the users to memorize the dependency structure of the specification, thus greatly reducing cognitive workload.
- 2- It gives the users an idea of the size and complexity of the specification.
- 3- It provides a “gestalt” overview of the specification, where properties and patterns can emerge to the users.
- 4- It displays elements as manipulable object on the screen which allows the users to cognitively “actualize” elements of the specification and visualize them in their operational context. This should foster the creation of a mental model of the system.
- 5- It separates the information about the high-level structural dependencies of the system from the information about the low-level behavior of elements, thus dividing the specification in two hierarchical levels of information, which makes it easier to manage intellectually. In SpecTRM-RL, the structural information is presented along with, and at the same level as the detailed behavioral information, in a single formatted text document where hyperlinks are used to show the structural connections between elements. Although the information contained in a SpecTRM-RL model can be divided in two hierarchical levels, the amount of information contained in large models may become overwhelming for the users and a feeling of disorientation can occur.
- 6- It provides an easy way to navigate the specification, both laterally (within the structural overview) and vertically (between the structural and behavioral information).
- 7- It provides simplification tools such as specification filtering and slicing to simplify the model and make it easier to extract information pertaining to the task at hand. Such tools are required because the complexity of the specification creates a need to simplify or create abstractions to a level where the model becomes intellectually manageable.
- 8- It uses perceptual cues such as color, shape and position to encode critical information about the model elements and their interdependencies. Purely textual specifications have a tendency to equalize the relative importance of information contained in specifications. They also fail to take advantage of some perceptual cues that can be used to highlight the most important information.

Expected Disadvantages

Our experience demonstrated the usefulness of the structural overview with the specification of the Vertical Guidance Annunciation Process system, which is of a size representative of real complex engineering systems. This function of the FMS includes more than 60 elements (State Variables, Devices, Macros, etc...) and more than 200 inputs and outputs. The scalability of the structural overview to larger systems is our primary concern. Visual occlusion is one of the main problems associated with the use of the structural overview with very large systems. The structural overview provides ways to alleviate this inconvenient but a size limit may exist above which the displayed structure becomes simply unmanageable.

Another problem with the structural overview is that human intervention is required to organize the computer-generated layout. This may be a good or a bad thing. By organizing the layout based on their understanding of the system, users may promote the creation of a high-quality mental model of the system. However, one-time users, for example, should be able to benefit from the layout organization of experts such as the creators of the specification. The structural overview allows each user to save and retrieve their favorite layout of the system structure.

Many algorithms exist for automatic layout generation [2] but they are difficult to use and have had limited acceptance. Consequently, it was decided to let the users or specification experts organize the structural layout based on their understanding of the system.

3.1.2 Question-Based Decision-Tree – (V2)

The Question-based decision-tree was created to help us understand the logic behind the most complex transitions in the part of the MD-11 specification. We found that it was easier to answer particular questions if the set of conditions in the transitions (which can be very large) are shown in sequence rather than in parallel. A decision-tree for each state variable seemed most natural in order to accomplish this goal.

Description

A sample decision-tree for the state variable *Active Add Drag Scenario* can be found in figure D. This state variable indicates which of the five possible scenarios for extending the airbrakes is active, if any. From left to right, each column represents one of the decisions that must be made to determine which transition will be taken, based on the state or value of the component of the model shown at the top of the column. The final states to which transitions can be made appear as leaves of the tree, at the right end of each branch. Usually, there are several ways to transition to a particular state; in that case, several tree leaves will bear the name of that particular state. The state *None*, in Figure D is an example of this.

Although more visually appealing and easier to manipulate, the basic tree does not bring much new information compared to other representations like AND/OR tables. However, by rearranging the information in a tree form, it appeared that each decision could be associated to an informal but explicit question whose answer is determined by the state of the formal element. This question is written at the very top of each column. One of the innovations here is to associate these two complementary pieces of information – the easily understandable informal question and the state of the formal element – in the same representation. The objective is to improve the understanding of formal logic by the human’s mind, whose mechanisms are informal. Each isolated decision can thus be viewed informally (“Is the Aircraft Overspeeding?”) or formally (“Is $ADC\ CAS > Active\ Descent/Approach\ Segment\ Predicted\ Airspeed + 5\ knots$ ”).

In order to demonstrate the functioning of the decision-tree, consider Figure D. Suppose the following information is known about the environment of the system:

- Input *Operational Procedure* is **Descent Path**
- Input *Active Descent/Approach Segment Thrust Type* is **Idle**
- Input *Active Descent/Approach Segment Speed Type* is **Mach**
- Input *ADC Mach* = **0.82**
- Input *Active Descent/Approach Segment Predicted Mach* = **0.80**
- Input *Flightphase* is **Descent**.

Based on this information, it is possible to identify which branch of the tree will be taken, from the root of the tree to the leaf labeled *Scenario 3*. That is, based on the information available, the state variable *Active Add Drag Scenario* would transition to the state *Scenario 3*.

It should be mentioned that the state transitions specified by the decision-tree are independent of the original or source state. If a transition depends on the current state of the state variable, the variable itself will appear in one of the columns and a path will be taken based on the current value of the state variable. This is not an issue if the system specification is robust and deterministic, in which case only one possible state will be reachable given any combination of input values [14].

The decision-tree representation of transition conditions is very concise when compared to textual or tabular representations. In fact, the decision-tree provided in figure D is equivalent to five pages of text or six AND/OR transition tables. Although conciseness is a desirable property of specifications, it should never have priority over readability. In the decision-tree representation, readability is increased by using perceptual cues such as color, fonts, and position to encode essential information about the specification behavior. As in the AND/OR tables representation, the order in which conditions are evaluated is technically irrelevant. That is, the columns of the tree can be reordered without affecting the end result. Every different column order is associated with a specific decision-tree layout. The interactive visualization computes and generates the tree layout based on the user-defined decision order. It would be possible to minimize screen space utilization by using algorithms to identify which of the possible decision orders will yield a more compact decision-tree. However, there is no assurance that the decision order associated with a more compact decision-tree will be more intuitive and easier to use, so the order is left for the users to decide. In theory, it would be possible to combine, or take the cross product of many state variables and generate larger decision-trees depending only on inputs to the system. However, such practice would quickly run into space utilization problems and different visualization strategies would have to be used to mitigate these problems.

Figure E presents a decision-tree generated for one of the most complex state variable of the MD-11 Vertical Guidance System. It is equivalent to 20 pages of textual specifications or 12 large (up to 21x28) SpecTRM tables. The sheer size of this tree makes it difficult to extract information and understand the behavior of the state variable. However, the visualization allows

the user to perform behavioral slicing based on user-defined scenarios in order to simplify the decision-tree. Behavioral slicing of formal specifications was identified by Heimdahl [15] as an effective way to manage complexity by extracting the information affecting the selected state variables. The behavioral slicing tool available in SpecTRM-RL was created based on Heimdahl's work and experience with using slicing on large specifications.

The interactive behavioral slicing tool provided with the decision-tree visualization has the effect of pruning the tree based on the value of some of the formal elements used in the decision-tree. The effect of behavioral slicing is shown in figure F, where the decision-tree for the state variable *Origin of Econ Airspeed Target* was sliced based on a scenario where: "The *Flightphase* is *descent*, but the aircraft is still above the normal trajectory, thus the current *Operational Procedure* is *Late Descent*". Branches of the tree that are unreachable based on the active slicing scenario are grayed out and compacted, which prunes them from the current visualization. Tree pruning provides more space and visual emphasis on the reachable branches, creating a visual foreground/background effect similar to that created with structural slicing in V1. Slicing scenarios are usually defined by restricting input values. However, restrictions on internal elements can also be used.

Although the slicing tools provided in the decision-tree visualization and in SpecTRM-RL are based on the same theoretical foundations [15], one of the major differences is that slicing in SpecTRM-RL is applied on the model as a whole, while slicing on the decision-trees is local and is automatically removed, unless specified, when the users change focus to another part of the specification. This feature was implemented to make it easy for the users to perceive the existence of an active slicing strategy. In the future, behavioral slicing on the specification as a whole could also be implemented in the visualization environment and offered to the users through a special menu.

Expected Advantages

Our experience has shown that many benefits are derived from the use of decision-tree representations of transition conditions:

- 1- Informal annotations coupled with formal notations help the users in understanding the reasoning involved in the decisions underlying the contents of the transition conditions. Such a combination of formal and informal representations is especially useful for novice users because not only does it make it easier to understand the behavior of the system, but it also helps novice users in getting familiar to formal notations.
- 2- Perceptual cues such as color, font, and position are used to encode critical information about the system behavior, thus reducing the user's cognitive workload.
- 3- The information is presented together in one concise notation. This helps identify patterns, make comparisons, and detect omissions.
- 4- Concise notations reduce the need for constantly switching between one part of the specification to another, thus reducing information fragmentation and navigation disorientation.
- 5- Transition conditions reordering provide the flexibility necessary for users to decide which conditions should be evaluated first based on their understanding of the system's functioning.
- 6- Interactive slicing allows for rapid local simplification of the specification based on the operational context of the task to be performed.
- 7- Out-of-focus information remains when using slicing in order to remind the users of the existence of some active simplification strategy.

Expected Disadvantages

Scalability to complex transition behavior is arguably the most important issue that had to be addressed when designing the decision-tree representation. Although more concise than textual or tabular representation, the decision-tree notation has to tackle difficult screen space utilization issues. In fact, it is believed that screen space utilization is one of the most important factors affecting the scalability of the decision-tree notation to complex transition behavior. Another disadvantage of the decision-tree notation is that human intervention is often necessary

to refine the computer-generated layout. More powerful layout algorithms could be used to improve the readability of the text areas and conditions, but users would most likely still have to slightly resize columns or text areas in order to obtain an easily readable layout.

Another factor affecting the scalability of the notation is the user's cognitive limits, which may be exceeded when using large decision trees such as that presented in figure 3-4 without simplification strategy. It should be mentioned, however, that textual and tabular representations run into even worse scalability issues. In fact, the scalability difficulties encountered by other representations were the single most important factor driving the creation of the decision-tree representation. A related thesis by Viguier [38] provides an informal comparison between the decision-tree representation and two existing, well-accepted representations (textual and tabular) based on the comparative framework defined by Zimmerman [40]. Another anticipated drawback is the impression of time sequence conveyed by the decision-tree representation. This illusion is a secondary effect of the representation of transition conditions in sequence, rather than in parallel. Novice users may be confused by such a time-order illusion, and it may distort their mental model of the functioning of state machines.

3.1.3 State Transition Diagram and Inversion – (V3)

Figure 3-2 shows a visualization of a typical circle-and-arrows state transition diagram that was augmented with an inverse state transition diagram in which the information in the original diagram is redundantly encoded in order to assist in answering different types of questions.

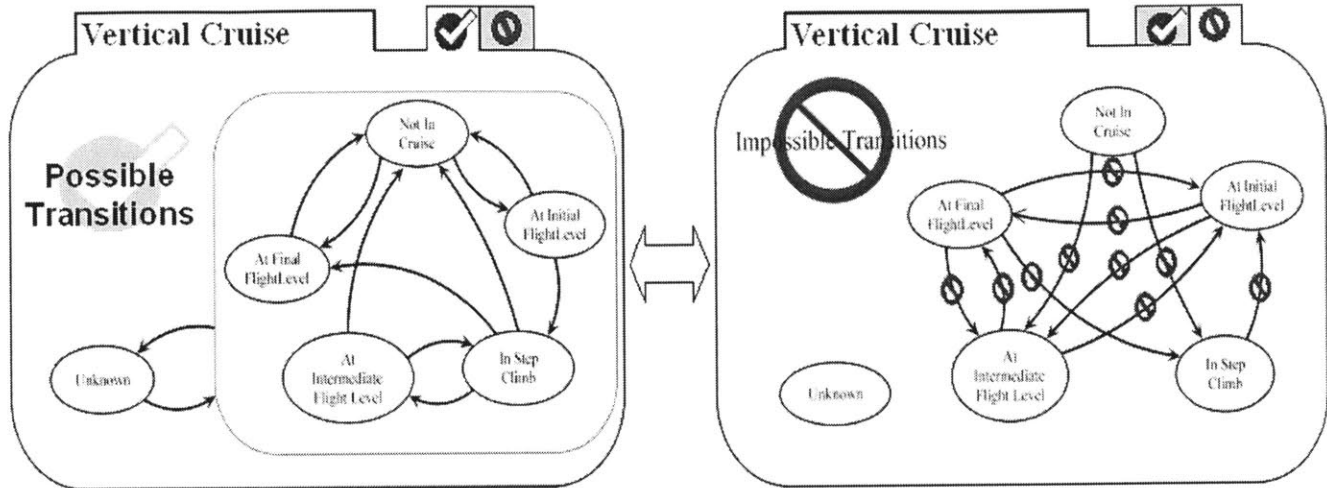


Figure 3-2: A sample state-machine transition diagram and inverse transition diagram taken from the MD-11 vertical guidance specification. This state variable describes the vertical attitude of the aircraft during cruise.

Motivation

A state-machine representation such as the cruise control system presented in figure 2-1 provides a complete, easy-to-understand representation of the behavior of a simple system. However, for larger systems, such a representation quickly runs into major scalability problems. Complex transition conditions are too large to be displayed on the transition arrows, and it is simply impossible to display the entire state machine of complex systems including thousands of state values. Nevertheless, users of state-based specifications often find it useful to see part of the flattened state machine using a traditional state transition diagram. Such a representation is useful because it explicitly shows the possible state transitions, both into a state, and out of a state, which is impossible to do using tables or decision-trees. Even though it is impossible to display the state-machine of entire systems, it is possible to display the state-machine resulting from the combination of a few selected state variables of the system. In this case, the total number of state circles will be equal to the product of the number of states in each state variable.

It was also found that the inverse state transition diagram is better suited to some particular tasks than the traditional state transition diagram. For example, if a user is interested in system safety and wants to make sure that no undesired transitions will take place, a representation of the impossible transitions of the state machine may be more appropriate to the task than a

representation of the possible transitions. Moreover, in many complex systems, every state is reachable from almost every other state; so displaying the impossible state transitions may be more relevant. Consequently, the state transition diagram can be displayed either in its traditional form or in its inverted form (figure 3-2).

Description

Figure 3-2 shows an example of a state transition diagram for a state variable of the MD-11 vertical guidance system. Both the traditional state transition diagram (left hand side) and the inverse state transition diagram (right hand side) are displayed in this figure. The arrows between states represent possible transitions in the state transition diagram, and impossible transitions in the diagram in inverse state transition diagram. The information in the inverse state transition diagram is the same as that in the traditional diagram, but it is recoded to make it easier for the user to perform particular tasks. The state transition details are not shown in this visualization but may be accessed through a simple click on a transition. States that are reachable through every other state are displayed as being connected to a contour including the entire state-machine. The state *Unknown* in figure 3-2 is an example of such a state. A single state variable is displayed in figure 3-2, however, as mentioned previously, it would be possible to take the cross-product of multiple state variables and to display the resulting state-machine in a similar way.

Expected Advantages

This visualization is arguably easier to use, and more intuitive than V1 and V2. It provides useful additional information that could only be extracted from the specification at the cost of considerable analysis efforts. In addition, it scales relatively well to complex state variables, but its usefulness is limited to the visualization of two or three combined state variables, after which the resulting state machine becomes unmanageable.

3.2 Taxonomy of Visualizations

This section introduces a taxonomy made of six dimensions upon which visualizations for formal requirements specifications can be classified. These dimensions are not exhaustive. Existing dimensions could be modified or more dimensions could be added to fit a particular purpose. However, the taxonomy presented is a good starting point and proved to be very useful in discussing and classifying the visualizations created.

3.2.1 Scope

The visualization may focus on the structure of the model or the goal may be to visualize the behavior of the specified system.

	V1	V2	V3
Scope	Structure	Behavior	Behavior
Description	Visualization based on dependency links between the elements of the model.	Visualization displays transition conditions for a single state variable.	Visualization displays possible transitions for chosen state variables.

3.2.2 Content

The visualization may include the entire model, perhaps using a different notation (e.g., symbolic, tabular, or graphical), or information may be elided. Elision is the ability to temporarily hide parts of the specification that are not of immediate interest. When information is elided, it may still be useful to retain some of the omitted information as context, but it is grayed out or somehow denoted as background rather than foreground. Alternatively, the visualization may not provide context beyond the information provided in the visualization itself.

	V1	V2	V3
Content	Elided	Elided	Elided
Description	The visualization is based on entire model but details are elided for readability.	The visualization is created based on a single chosen element of the model.	The visualization is created based on a single chosen element of the model.

3.2.3 Selection Strategy

The visualization may be created through *slicing* the basic formal model, i.e., a selection based on dependences between the parts of the model or by *filtering*, i.e., eliding parts of the model based on a common property or attribute.

	V1	V2	V3
Selection strategy	Filtering/Slicing	Manual/Slicing	Manual
Description	The behavioral information is <i>filtered</i> to display only structural information and interactive <i>slicing</i> can be used to simplify the visualization.	The users select the element they desire to visualize and interactive slicing can be used to simplify the chosen behavioral visualization.	The users manually select the state variable they wish to visualize based on the task they wish to accomplish.

3.2.4 Annotation Support

The visualization may include only information provided in the original specification or the user may be able to add extra domain knowledge through annotation.

	V1	V2	V3
Annotation Support	Not supported	Supported	Not Supported
Description	The visualization is created solely based on the information contained in the formal model.	Informal questions may be added to the visualization to support the understanding process.	The visualization is created solely based on the information contained in the formal model.

3.2.5 Support for Alternative Search Strategies

Visualizations may be provided that supports a particular search or problem-solving strategy without any options for the user. Alternatively, the user may be able to specify the search strategy to be supported by the visualization. A third option is to provide interactive visualizations where the user can change the search strategy while navigating through the model.

	V1	V2	V3
Support for Alternative Search Strategies	High	High	Low
Description	Search strategies include top-down review, interactive slicing, filtering, grouping, etc...	Search strategies include interactive slicing, re-positioning, and condition re-ordering.	Limited search strategies.

3.2.6 Static/Dynamic

A static visualization is a snapshot of the specified behavior of the system at a particular time or a static description of all possible behavior. Dynamic visualizations or animations show the specified behavior of the system as it changes over time.

	V1	V2	V3
Static / Dynamic	Static	Static	Static

Chapter 4

Requirements Specifications Visualization Design Principles

Not all visualizations are useful. Sometimes they may even be misleading. In fact, it appears that visual representations have more potential for “going wrong” than symbolic representations. Text is more constrained by nature. Its linear character provides cues to the reader even when its format is dysfunctional [24]. Visual representations provide more freedom at the cost of a greater potential for misinterpretation and confusion. A careful evaluation of the visualizations created for reviewing and understanding formal specifications is necessary to maximize the usefulness of visualizations while minimizing the potential for misleading users. This chapter presents nine principles to be used for evaluating potential visualizations and for creating effective ones. These principles were either adapted from research in related fields such as visual programming and human-computer interactions, or created based on previous user studies and on our experience in specifying complex systems [7].

4.1 Minimize Semantic Distance

Semantic distance is a concept devised for human-computer interface design to describe the distance between the user’s model of how the system works and the model of the system presented by the user interface [16]. In the context of visualizing formal specifications, semantic distance is the distance between the model in the system specification and the mental model of the system in the mind of the users of the specification. Readability and reviewability will be enhanced if visualizations are provided that minimize semantic distance. Professor Leveson has found informally that reducing the semantic distance between standard engineering models of complex systems and formal specification notations can increase acceptability and usability of formal specification languages among people in industry who previously rejected out of hand the use of such specification languages. As an example, the formal model of TCAS II (a collision avoidance system for commercial aircraft), written with Professor Leveson’s modeling language, has become the official specification of this system [22].

A previous experiment on the readability of various notational features compared the specification of the conditions on state transitions using text, tables, graphical logic gates, and propositional logic. Every computer science student in the experiment commented on the difficulty of using the logic gate notation while the engineering students were mixed [41]. However, similarity to standard notations is not the only relevant criterion, as every participant in the experiment preferred the tables and made the fewest errors in using them. Moreover, even computer science students, well trained in the use of propositional logic ranked this notation at the bottom for both of these criteria. This indicates that users are willing to use new notations if they believe it to be better, or more effective. These results confirm our industrial experiences.

Specifications of complex systems are often very large and it is practically impossible for a single person to have a complete understanding of the functioning of the specified system. Consequently, each specification user's mental model of the system will be different even though the content of the specification does not change. Matching a visualization to a specific task is beyond the scope of this work, mostly because of the almost infinite amount of possible task-visualization combinations. Because it is not possible to have a perfect visualization-task match, visualizations of formal specifications should provide customization tools that will allow the users to "shape" the visualization according to their mental model of the functioning of the system or according to the specific task they have to perform. As such, a graph-like display of the structure of a formal specification such as V1 should provide customization tools that allow users to move elements around and group them together according to the user's understanding of the system's functioning. In some cases, providing a difficult-to-use, computer-generated structural layout of the specification instead of a pre-organized layout in order to force users to organize the layout based on their understanding of the system could be a very effective way to promote the creation of a mental model of the system among novice users. Decision-tree visualizations of transition conditions such as V2 should allow the user to decide the order of the sequence that will be used to evaluate the conditions based on their understanding of the system's functioning. Since the customization of a visualization has no effect on the underlying formal model upon which the base visualization is built, users should be encouraged to use those tools in order to shape the visualization to their own mental vision of the system, instead of using the default visualization provided.

4.2 Match the Task being Performed

Gilmore and Green's basic *match/mismatch hypothesis* states that problem-solving performance depends on whether the structure of a problem is matched by the structure of a notation [10]. Applying this hypothesis to visualization implies that the most effective visualizations of requirements specifications will be those that most closely match the problem being solved or task of the specification user. The goal is to match the task to be performed with a visualization that minimizes the amount of cognitive processing required to perform the task.

As mentioned previously, it is not possible to provide visualizations for every potential task to be performed by specification users. However, a few tasks are commonly performed during the review of specifications such as searching for information and navigating between overall structure view and detailed element information. Visualizations should support the most common tasks first. As such, V1 provides a structural overview of the formal model that can be simplified depending on the context of the task to be performed and that provides seamless access to lower-level detailed model information.

4.3 Support the Most Difficult Mental Tasks

Some tasks that use formal specifications will be more difficult than others in terms of the number and difficulty of the cognitive processing necessary to perform those tasks. The most useful visualizations in this context will obviously support the hardest tasks and not simply those that are easiest to create or are appealing to the visualization tool builder. This principle implies that the first step in creating useful visualizations is to determine who the users will be, to perform a task analysis of their potential uses of the visualization, and to analyze the difficulty of performing the task without a special visualization of the formal model.

V1 was developed to overcome a difficulty in creating a mental model of the complex dependency structure of a formal model when using hypertext representations. V2 was developed because even the most readable notations for the specification of transition conditions run into scalability issues in the case of highly complex system behavior. The table notation, for example, works great for a few possible states with reasonably complex transition conditions. However, cognitive workload increases quickly when the behavior of the system becomes complex and the tables become larger. As an example, one of the state variables used in the

specification of the MD-11 Flight Management System has seven possible states, each of which has associated transition condition tables reaching a size of twenty rows by thirty columns. Such complex transition conditions are impossible to manage intellectually and require visualizations that reduce the user's cognitive workload.

4.4 Highlight Hidden Dependencies and Provide Context when Needed

Blackwell and Green [11] define a hidden dependency as “a relationship between two components such that one of them is dependent on the other, but that the dependency is not fully visible”. Any notation makes some dependencies clear while obscuring others. These hidden dependencies may or may not be important in performing a particular task. If the dependencies are relevant to a user, then visualizations should be provided that perceptually highlight those dependencies. A good representation will in general point out dependencies or show causal relationships between different automation behaviors. For example, some formal specification languages based on state machines organize the specification in such a way that it is easy to determine the previous states but not the potential states that follow the current state and vice versa. For example, tables easily show which states will be accessible, given a current state (“going-to” perspective). However, much effort is needed if it is desired to know which states can transition to a current state (“coming-from” perspective). It is possible to overcome such difficulties by using visualizations such as V3 that graphically show all possible arrival and departure combinations of transitions between states.

Another example of hidden dependency often encountered in requirements specifications is the indirect relationship between elements that are not visible when using a pure hypertext representation of the specification. V1 was developed in order to be able to easily visualize indirect relationships between elements. Non-structural dependency links can also be user-defined if, for example, element B does not depend explicitly on element A, but element A has to be informally taken into account when reviewing element B.

Since formal specifications of complex systems contain a large amount of information, only a small part of the information will be used at once to construct visualizations. In cases where only a small part of a specification is being displayed, context has to be provided for the rest of the specification if it is required for the task at hand. Context information is very important in visualizing formal specifications for many reasons. It is used to remind the user that only a small

part of the specification is displayed and to clearly indicate which part of the specification is displayed in context. While displaying context information is important, efforts should be made to minimize spatial disorientation when changing the information in focus. Spatial disorientation can occur if a change in information in focus is accompanied by a sudden change in visualization layout. The display of context information was very carefully implemented in V1 and V2 using a “background-foreground” approach. For example, if slicing is used on V1 or V2 to select a small part of the available information, the selected information is put in focus by being enlarged (“pulled” to the foreground) while non-selected information is put in context by being greyed out and compressed (“pushed to the background”). Figure C provides an example of information being put in focus-context by using an input-to-output slicing selection strategy on the state variable *Origin of Level T_D*.

4.5 Support Top-Down Review

Graphical overviews of the entire specification can be very powerful. During the review sessions of a formal specification of TCAS-II provided for the FAA, Leveson observed that domain-expert reviewers would spend hours discussing a simple graphical overview of the state variables and state values (see Figure 4-1) used in the specification without referring to any information about the conditions on the transitions between the states, which were not visible in the graphical overview. Expert reviewers prefer to start with a high-level overview of the system state values before delving down into the details of the transitions even though all the information in the overview could be deduced from the structure and content of the rest of the specification. This is an example of the *gestalt* effect in cognitive psychology in which providing an overview makes overall structure or relationships visible or clearer [24]. V1 was designed to provide a high-level structural overview of the system that allows the users to easily access lower-level system behavior information such as that provided in V2 when needed.

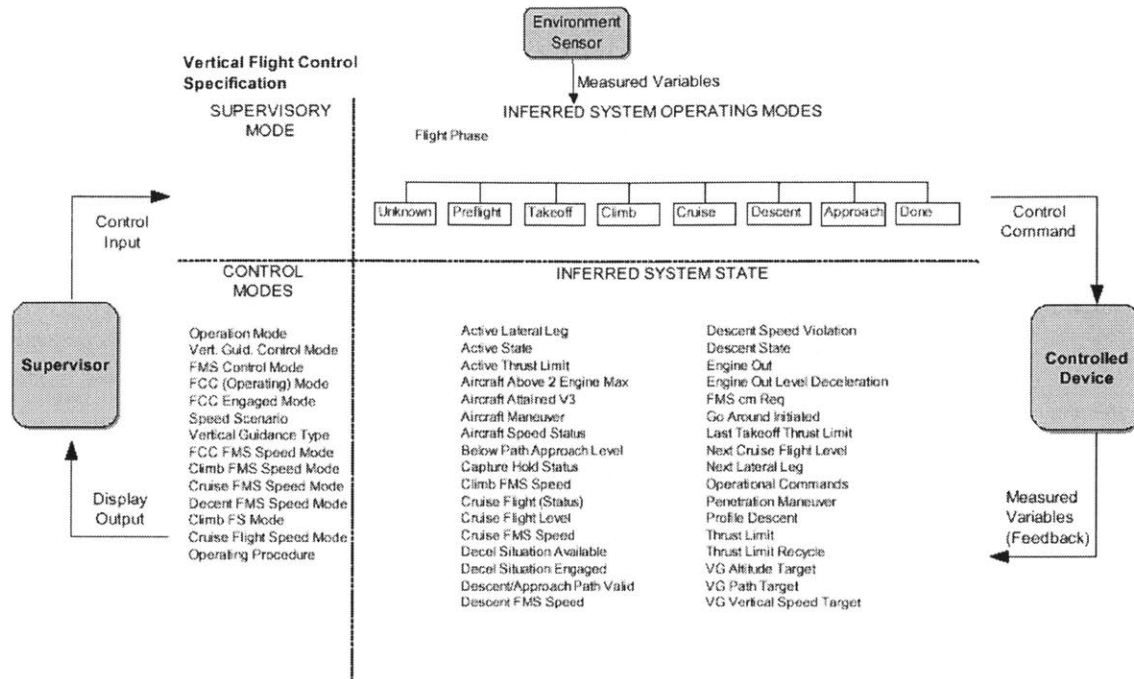


Figure 4-1: Sample graphical overview of the modes and state variables of a digital avionics system.

4.6 Support Alternative Problem-Solving Strategies

A principle of cognitive psychology is that the reasoning paradigm is distinct from the representation paradigm. "The cost of reasoning about a particular representation may vary, depending on how the programmer's reasoning shifts" [1]. Therefore, the representation may need to change as the user's reasoning process shifts. In addition, different people will employ different problem-solving strategies for the same problem. Experts are more likely than novices to change strategies while problem solving and to exhibit flexibility in their strategies [24]. Supporting expert use of formal requirements specifications with visualization will require supporting flexible search strategies and the ability to navigate between abstract and detailed views, as well as within detailed views. This principle emphasizes the fact that there will not be a fixed set of visualizations that are best for all people solving the same problem or performing the same tasks.

In order to accommodate users of different backgrounds and expertise levels with powerful tools that will be useful in solving a large variety of tasks associated with the review of formal specifications, much flexibility must be built in the visualizations. For example, V1 offers many

different ways of selecting the information to be put in focus (slicing, filtering, etc...) and V2 offers different ways of re-ordering and simplifying the behavioral information in order to solve many different tasks. Building flexibility in the visualization is important to allow the users to customize the visualization to their way of thinking about the system and/or to the specific tasks they wish to perform. However, building flexibility in the visualization may increase the initial workload of novice users. It creates a need for more user training with the visualization tools, and may delay the benefits of the adoption of powerful formal specification tools, just like the adoption of a powerful CAD system in a company would initially increase the workload and decrease the throughput.

4.7 Show Roles Being Played

Visualizations should provide insight into the role being played by a specific part of the specification. As an example, consider the use of modes in control system requirements specifications such as those used in SpecTRM-RL [22]. Modes are a common way of abstracting and grouping important subsets of behaviors of the overall system behavior in control systems. That is, modes divide the overall system behavior into a set of disjoint behaviors, e.g., the behavior of the flight management system during landing mode or cruise mode. Modes are useful in simplifying (reducing) the amount of specified behavior that must be considered at any time. If there are multiple independent mode classes, the system behavior may be described in terms of the cross product of the individual mode values. Basically modes allow us to divide the behavior of the system into non-overlapping chunks that are easier to process cognitively.

Multiple modes allow chunking on different dimensions. Visualizations for control system requirements specifications should allow identifying and highlighting the role of each mode in the overall system behavior being described and the role played by each of the components of the specification in a particular mode. Similar advantages accrue to expressing other important roles in requirements specifications.

4.8 Provide Redundant Encoding

Any representation makes some questions easier to answer while making others harder [8]. For example, a list or table showing classes taught, time, and professor that is ordered by class

number will make it easy to answer questions about who is teaching a particular class, but much more difficult to answer a question about which classes a particular professor is teaching. A different ordering will make the latter question easier to answer than the former. Casner's task-analytic approach suggests that the effectiveness of any visual display will depend on both the type of task to be performed and the cognitive processing required to perform the task [4]. The objective is to match the task to be performed with the representation that minimizes the cognitive processing necessary to perform the task. Since it is not always possible or practical to perform a task analysis on the specification users, a small number of representations built upon the same underlying formal model should be available for users to choose the representation that fits best the task they have to perform.

In addition to trying to provide visualizations that will be well adapted to perform particular tasks, visualizations need to consider the large variety of users involved in the review of complex specifications. Previous experiments have shown that while it is possible to design representations that will be nearly optimal for a group of users, those representations will usually be sub-optimal for a different group of users [41]. By providing redundant but different encoding of the same information about the required behavior of the software, support can be provided for a variety of user groups performing a variety of different tasks.

V1 and V2 were developed based on the underlying hypothesis that multiple notations and visualizations generated from a common formal model will improve the requirements review and understanding process. In fact, V1 and V2 provide redundant encoding of the formal models initially constructed using SpecTRM-RL. V1 provides redundant encoding of the structural information contained in the SpecTRM-RL model, while V2 provides redundant encoding of the behavioral information contained in the AND/OR transition tables of the SpecTRM-RL model.

4.9 Show Side Effects of Changes

Requirements specifications of complex systems contain a large amount of information and usually demonstrate high coupling and interdependencies between elements of the system. Consequently, side effects of changes will propagate throughout the specification and it may be difficult to ensure that every side effect of a change has been identified and properly addressed. Because of this, visualizations should allow investigating the impact of a change in one part of a

specification on other parts. In other words, visualizations should explicitly show the indirect effects of changes on the rest of the specification.

This list of principles is not exhaustive. Depending on the application, other design criteria may have to be taken into account. In some cases, some principles presented in this chapter may overlap or conflict and tradeoffs may be needed. However, it is believed that these principles provide an excellent starting point for the design of effective formal specification visualizations.

Chapter 5

Experiment Design

The principles proposed in the preceding chapter have been adapted from other fields or introduced on the basis of our specification experience [7]. They should not be considered as a rigid and exhaustive set of rules but as a starting point. They will need to be refined and evaluated against a variety of visualizations. Although the visualizations described previously were useful in understanding the MD-11 specification, this anecdotal evidence does not prove their usefulness to a broad class of users and specifications. The present chapter describes the design of an experiment with human subjects intended to be the first step into validating the application of the principles to formal specifications.

5.1 Experiment Objective

The experiment described in this chapter has two major objectives. The first objective is to evaluate formally the usefulness of the Structural Overview (V1) and the Decision-Tree (V2) visualizations for the review of formal specifications. The second objective is to informally evaluate the validity of the principles presented in the preceding chapter. The visualizations used in this experiment (V1 and V2) rely heavily on four principles we selected as the most interesting to evaluate. Those selected principles are:

- Highlight hidden dependencies and provide context when needed
- Support top down review and provide “gestalt” overview
- Support alternative problem-solving strategies
- Provide redundant encoding

It is important to understand the difference between the two objectives. The preceding chapter introduced a set of principles that should be used as a guide to the creation of formal specifications visualizations. However, the design of effective visualizations is an inherently organic process requiring much creativity from the designer. Such a creative process is difficult to quantify and does not lend itself to the realization of a fixed set of guiding principles. Consequently, the resulting visualizations use the guiding principles in an abstract, interlaced manner and it is impossible to isolate or extract individual principles from the visualizations.

An exhaustive evaluation of each of the guiding principles separately would require the use of visualizations where a single principle can be removed in order to compare the performance of subjects using the base visualization with that of subjects using the “reduced” visualization. Creating visualizations where individual principles can be removed would probably not be feasible. Even if it was possible to obtain such visualizations, the design of suitable experiment questions would be much more difficult because one would have to very carefully assess the side effects of removing a single principle on the information contained in the visualization. In fact, it would be impossible to know whether the differences in performance are a result of the principle removal or of the completely different visualization obtained from removing the principle.

A different approach will be taken in which the principles will not be validated individually. Instead, they will be validated through the evaluation of the visualizations they helped create. By comparing the performance of human subjects in specification reviewing with and without the use of visualizations, it is possible to indirectly assess the validity of the principles used to create those visualizations. However, since the principles are evaluated indirectly through visualizations, there will always be a doubt that the merit and format of the visualizations themselves may be responsible for the performance difference, whether or not the design principles were used. Consequently, more visualizations designed through these principles and more evaluations will have to occur in order to build confidence in the validity of the principles. This experiment is intended to be the first step in this direction.

5.2 Experiment Hypothesis

Based on the objectives of the experiment described above, the general hypothesis of the experiment was defined as:

The use of interactive visualizations created based on the proposed design principles will improve the formal specifications reviewing process.

In addition to the formal evaluation of the visualizations created, an informal assessment of the subjects' performance and problem-solving strategies will be conducted. It is believed that some of the most important experiment results will come from a careful observation of the strategies employed by users and by emergent problem-solving patterns.

5.3 MD-11 FMS Formal Specification

The formal specification used as a case study for this experiment was extracted from an experimental specification of the MD-11 Vertical Guidance System from Honeywell. The MD-11 is a three-engine airliner that was produced by McDonnell-Douglas/Boeing in the 1990s. This specification was originally written by Lance Sherry in 1989 and presents some characteristics that make it suitable to a translation into a formal specification. A SpecTRM-RL formal model of the Honeywell specification was started at SERL in 2000, and completed in 2002.

The Vertical Guidance (VG) is a subsystem of the MD-11 Flight Management System (FMS). Its role is to compute the altitude, speed, thrust and pitch targets necessary for the aircraft to follow its pre-established vertical flightplan. These targets can be sent to the autopilot or displayed in the cockpit as an advisory to the pilots. The Vertical Guidance function of the FMS is separate from the Lateral Guidance function.

One of the functions of the Vertical Guidance system is to provide visual feedback to the pilots about the state of the aircraft. For example, the Vertical Guidance system informs the

crew about the current system's operating mode, about the relative position of the aircraft with respect to its ideal trajectory, and about whether airbrakes should be extended to increase the aircraft drag. This feedback function of the Vertical Guidance system is known as the “*VG Annunciation Process*”. This function of the Vertical Guidance system was used as a case study for our experiment.

VG Annunciation Process sends outputs to most of the display units in the cockpits. This includes the Primary Flight Display (PFD), the Navigation Display (ND), and the Multifunction Control/Display Unit (MCDU) (Figures 5-1, 5-2). The VG Annunciation Process function also provides data to the flight control computer (FCC) for display purpose (Figure 5-3).

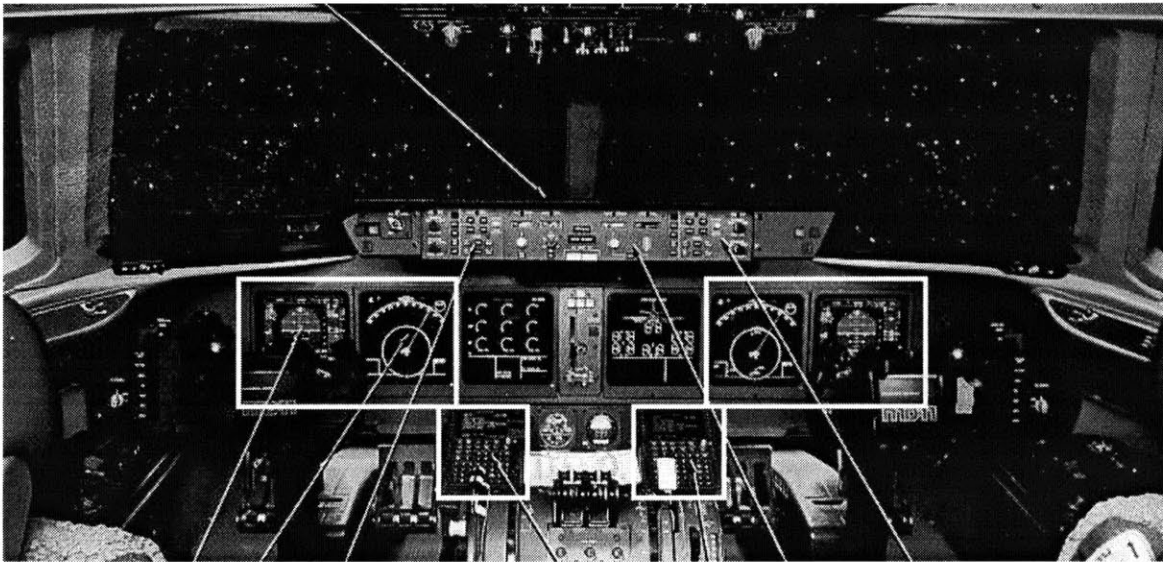


Figure 5-1: Inside view of the MD-11 cockpit. The VG Annunciation Process sends feedback information to the highlighted display units.

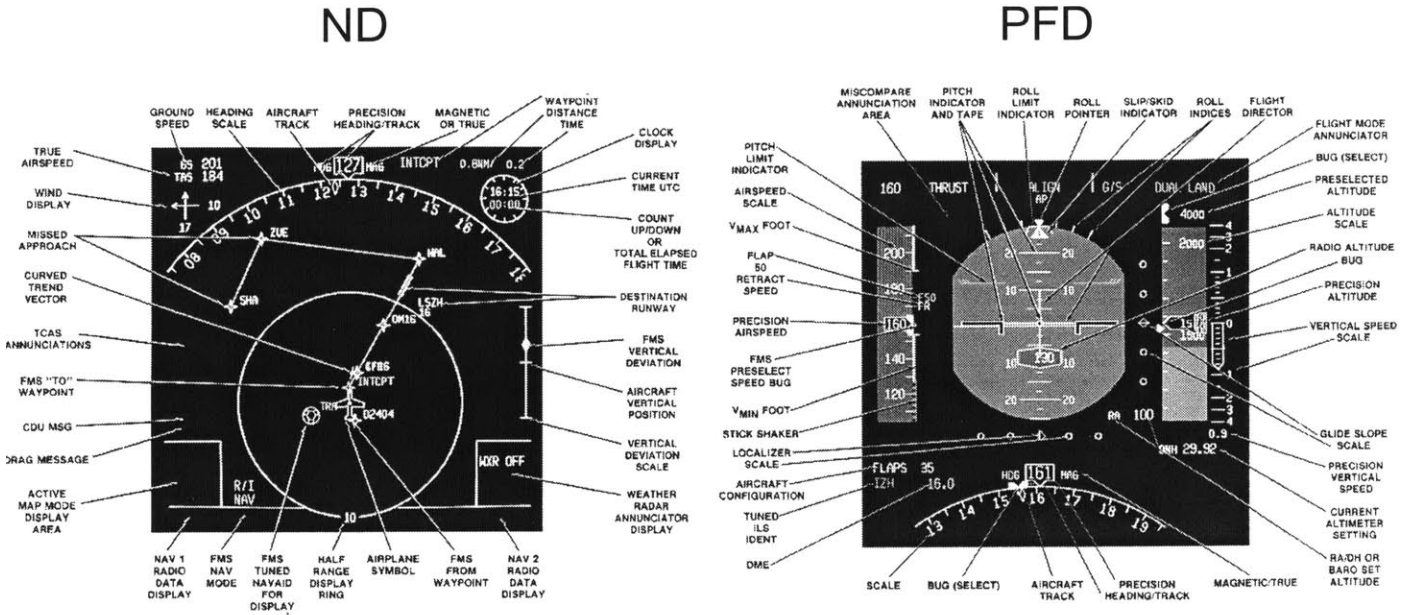


Figure 5-2: Detailed view of the main display units of the MD-11 cockpit.

The VG Annunciation Process function receives inputs from other processes of the Vertical Guidance system, other functions of the FMS (such as Lateral Guidance or Navigation), and from other systems external to the FMS such as the Flight Control Panel (FCP, Figure 5-3). The Flight Control Panel is the main flight control interface of the aircraft. The pilots use it to toggle the autopilot on and off, and to bypass the guidance functions by manually entering targets.

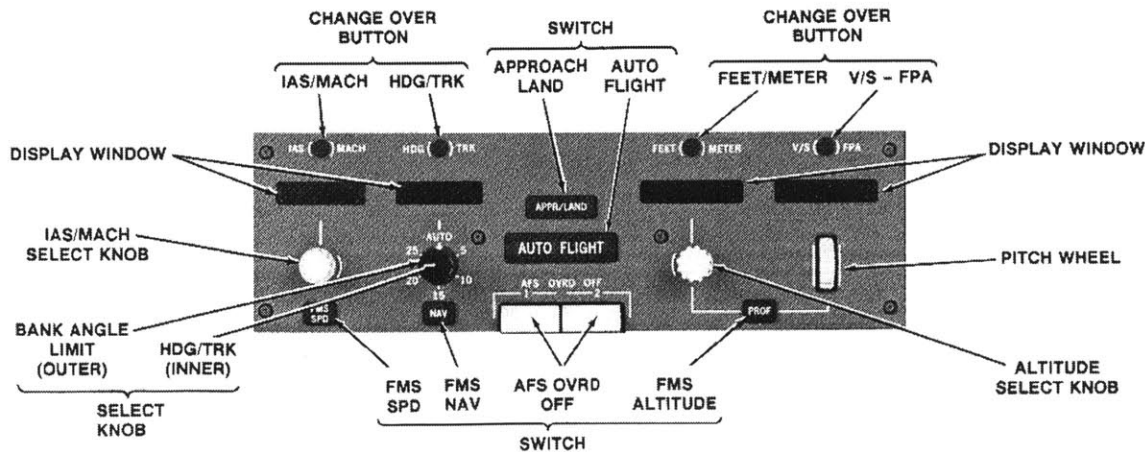


Figure 5-3: Detailed view of the Flight Control Panel (FCP) located on the glareshield panel of the MD-11 cockpit.

Figure 5-4 provides an overview of the VG Annunciation Process case study in its hierarchical context, along with its main Input/Output devices.

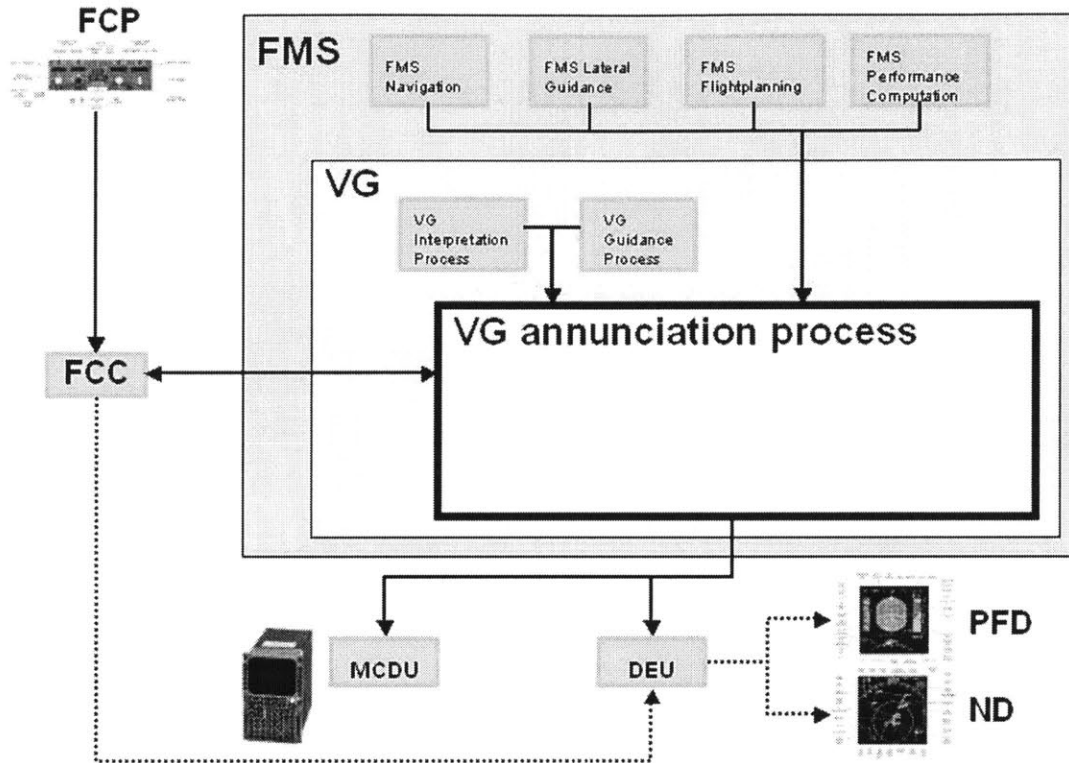


Figure 5-4: Hierarchical context of the Vertical Guidance Annunciation Process function along with its main I/O devices.

The state-based specification of this system is of a size representative of today’s engineering systems. The VG Annunciation Process system includes 36 state variables having a total of 170 state values, which adds up to a state-space of about 10^{18} possible states. The system also computes eleven continuous variables, and the system state depends on more than 120 inputs coming from 10 different devices. It took a few months for two graduate students to understand the original specification and extract the information necessary to create a SpecTRM-RL formal model out of it.

The VG Annunciation Process system is fairly complex. Its complexity is semantic (meaning and behavior of its different components), functional (several interdependent functions coexist), and structural (there is a high level of coupling between the various system elements).

This system was chosen because it is a real safety-critical system: A wrong feedback to the pilots can lead to hazardous situations. Also, it has the appropriate size and complexity for our purpose. It was possible for two engineers to implement it in a few months, yet it is large and complex enough that it is impossible to grasp entirely in the duration of the experiment.

However, there are some constraints associated with this choice. A certain level of domain knowledge is required to understand and use such a system. Since the experiment subjects do not possess this knowledge, we either had to design an experiment that does not rely on this domain knowledge, or provide the subjects with a very long tutorial. It is very difficult to design experiment questions that require no prior knowledge of the system, and an experiment based on these questions would have less value since reviewing a system specification is a task that usually requires some knowledge of the system's functioning. Consequently, we decided to provide the subjects with a relatively extensive tutorial of the specified system.

5.4 Tools Description

This section describes the user interface of SpecTRM and the visualization tool used in the experiment.

5.4.1 SpecTRM

Chapter 2 presented the principles that guided the design of SpecTRM. This section presents an overview of the SpecTRM GUI. For more detailed information about the functioning of SpecTRM, the interested reader is referred to the SpecTRM user manual [30].

Figure A provides a snapshot of the SpecTRM GUI. The SpecTRM GUI is made of two distinct panels. The left panel is called the "project browser". It is a tree similar to the file browser found in Windows Explorer. The project browser is used to navigate within the specification. The right panel is called the "project editor". It presents many characteristics common to a formatted text editor. The entire model is displayed and can be edited directly in the project editor panel.

As mentioned previously, a complete SpecTRM intent specification includes seven hierarchical levels that form a means-end hierarchy. The level 3 of SpecTRM defines the blackbox behavior of the system using a requirements language called SpecTRM-RL. Because it was not possible to extract the designer's intent from the paper specification used to create the model, only the externally visible behavior of the system was specified. Consequently, out of the seven hierarchical levels necessary to obtain a complete SpecTRM Intent Specification, only level 3 was used in the experiment.

Level 3 specifies the blackbox behavior of a system using elements of the following types: modes, state variables, macros, external devices, inputs, outputs, and functions. Other element types exist but were not used in the specification of our system. The different elements in a SpecTRM specification are specified sequentially and grouped by categories.

Any given element X includes two lists of hyperlinks that reflect its structural dependencies. All the elements that directly influence the behavior of element X appear in the "References" field; all the elements whose behavior is directly influenced by the value of element X appear in the "Appears in" field (see Figure A).

The behavior of elements such as state variables and macros needs to be explicitly specified in terms of possible state values and transition between those values. It was previously mentioned that SpecTRM-RL uses AND/OR tables for specifying state transition conditions. An example of such tables is provided in Figure G. Each element specification also includes fields for additional information such as element description, informal comments, and exception-handling behavior.

SpecTRM users navigate through the specification by using the project browser and by following hyperlinks. One way will usually be preferred over the other depending on the task at hand.

There are constraints associated with the size of the display. Only one element can be displayed at once in the project editor panel. This makes some tasks (such as comparing two different state variables) rather difficult.

5.4.2 Visualization Tool

The visualization tool is a GUI coded in Java using the Eclipse development environment. It displays V1 and V2 in two separate panels and allows interactions between the two. Figure H provides a snapshot of the Visualization Tool GUI. The upper panel (V1) displays the structural overview of the specification. The users can interact with V1 directly through clicking and dragging elements or indirectly through a pull-down menu that provides further options for simplifying and changing the layout of the visualization.

The users navigate the specification using the upper panel (V1), which can be considered as a map of the specification. The bottom panel displays behavioral information about the element selected by the user in the upper panel (V1). This panel is divided in two parts: the information panel (left-hand side), and the decision-tree panel (right-hand side). The information panel contains the same additional element information available in the SpecTRM model such as description and comments about the selected element. If behavioral information is available for an element selected in V1, the decision-tree panel will display the decision-tree (V2) associated with that element. The lower panel allows detailed information about a single element to be displayed at once. However, the users have the possibility to save previously viewed trees in new tabs for future use. Just as in V1, the users interact with V2 directly by clicking, dragging, and resizing objects on the screen. Behavioral slicing is available through simple mouse clicks. Just as with SpecTRM, some display size constraints are associated with the use of the visualization tool. For complex systems, it is usually impossible to display the complete structural overview and a large decision-tree at the same time. In order to overcome this problem, each panel is easily resizable to the extent where it occupies the entire screen space.

5.5 Experiment Methodology

This section provides practical information about the experiment itself, including the subject selection, question design, and analysis methods.

5.5.1 Subject Selection

Twelve subjects volunteered to participate in this experiment. All subjects were graduate students at MIT with backgrounds in either Electrical Engineering/Computer Science or Aerospace Engineering. The subjects were selected to ensure that they had little or no previous exposure to SpecTRM, the visualization tool, and formal specifications in general. The subjects had no detailed knowledge of the functioning of an airliner's FMS.

5.5.2 Tutorial

The first part of the experiment consisted of an hour-long tutorial where subjects are introduced to the relevant MD-11 systems and the tools used to answer the experiment questions. In order to obtain statistically relevant results and to compare problem-solving strategies, it was important to ensure that every subject had approximately the same level of knowledge and experience with formal methods, state machines, and digital avionics systems. Since it was not possible to expect all subjects to have the exact same expertise in these fields, we decided to reduce the differences by recruiting subjects with rather uniform backgrounds and by providing them with a relatively extensive tutorial.

The tutorial was in the form of a PowerPoint presentation including 3 parts:

1. A general introduction to the MD-11 Cockpit Displays and the Vertical Guidance system, along with a presentation of the V G A n n u n c i a t i o n P r o c e s s system and its Input/Output interfaces.
2. An introduction to formal requirements specification and state machines, along with a tutorial on SpecTRM-RL and the visualization tool. This part focuses on practical skills by explicitly demonstrating how to perform typical tasks using both tools.
3. A practice session where the experimenter helps the subject in answering a sample question of the same difficulty level as the experiment questions.

The objective of this tutorial was to provide every subject with the knowledge and experience necessary to answer the experiment questions.

5.5.3 Experiment Questions and Tasks

The second part of the experiment consisted of three sets of two questions that subjects had to answer by themselves. An experimenter was present at all time during the experiment but could only answer questions pertaining to the use of the tools. Every subject was presented with the same questions in the same order, but the tools used to answer the question were selected randomly. Table 5-1 shows the ordering of the question/tool combination used for the twelve subjects. At the end of the experiment, every subject had answered one set of questions using SpecTRM-RL only, one set of questions using the visualization tool only, and one set of questions where both tools were available.

Subject #	Question 1	Question 2	Question 3
1	SpecTRM-RL	Visualization Tool	Both Tools
2	Both Tools	SpecTRM-RL	Visualization Tool
3	Visualization Tool	Both Tools	SpecTRM-RL
4	SpecTRM-RL	Both Tools	Visualization Tool
5	Both Tools	Visualization Tool	SpecTRM-RL
6	Visualization Tool	SpecTRM-RL	Both Tools
7	SpecTRM-RL	Visualization Tool	Both Tools
8	Both Tools	SpecTRM-RL	Visualization Tool
9	Visualization Tool	Both Tools	SpecTRM-RL
10	SpecTRM-RL	Both Tools	Visualization Tool
11	Both Tools	Visualization Tool	SpecTRM-RL
12	Visualization Tool	SpecTRM-RL	Both Tools

Table 5-1: Tool/Question combination for the twelve subjects.

The questions were designed to evaluate the subject's capacity to understand both the structure of the specification and the behavior of the particular elements. Some emphasis was put on the detection of indirect relationships between elements, and on the analysis of the effect of these relationships on the functioning of the system. Each question includes two parts that can be answered independently. Although no time constraint was officially enforced on the duration of the experiment, subjects were encouraged to move on to the next question if they spent more

than 20 minutes answering a question part. A time period of 20 minutes was deemed sufficient to answer each question part. A copy of the experiment questions is provided in Appendix B. Each question has a single best possible answer.

5.5.4 Post-Experiment Analysis

Three metrics were used to evaluate the performance of subjects using the visualization tool either as a stand-alone tool or as a complement to SpecTRM-RL. The following table (5-2) summarizes the performance metrics and provides a list of the variables used.

Hypothesis: The use of interactive visualizations created based on the proposed design principles will improve the formal specifications reviewing process.		
Independent Variables	Measure	Variable Type
Subject Background	AA / EECS	Categorical
Display Type	SpecTRM / Visualization Tool / Both	Categorical
Task	3 different tasks randomly assigned	Categorical
Dependent Variables	Measure	Variable Type
Answer to Question	Score 0-10	Ratio
Answering Time	Minutes	Interval
Question Difficulty	Difficulty assessment score 0-10	Interval

Table 5-2: Variables used in the experiment

The experiment was designed such that all independent variable types are of the type *Categorical*, and all dependent variable are either of the type *Ratio* or *Interval*. Such a combination of variables is appropriate for statistical analysis such as T-Tests or ANOVA. Randomizing the tool usage ensures that statistical analysis results can be obtained despite the

relatively limited number of subjects. A higher number of subjects would simply increase our confidence in the results obtained.

This experiment method allows an objective evaluation of the visualizations created for reviewing specifications as stand-alone tools or as a complement to SpecTRM-RL. It was decided to use SpecTRM-RL alone as a basis for comparison because it has been used extensively in large specification projects and its value as a reviewing tool has been proven in many occasions.

The results have to be analyzed while taking into account external factors such as the background and expertise of the subjects, the design and difficulty of the experiment questions, and the relevance of the specification and questions chosen with respect to general reviewing tasks.

As a complement to the performance metrics described above, the subjects' behavior and strategies were monitored during the experiment. In particular, the experimenters were interested in answering questions such as: What tool was used when both tools were available, and why? What navigation or search strategies were used? What particular visualization features were used? What level of success was obtained for each feature? What confused the subject? In some instances, the experimenters had to explicitly ask informal questions to the subjects in between questions in order to clarify the subject's objectives or strategies. In these occasions, care was taken not to suggest strategies or provide performance feedback to ensure that no bias was introduced in the experiment. The objective of these questions is to precisely observe and record the subjects' behaviors and actions while performing the tasks, and to analyze them afterwards. Emerging behavior patterns will be discussed and, if possible, linked to the application of the principles used to design the visualizations.

The experiment sound and screen capture were recorded to assist the experimenters in this investigation. In order to alleviate the bias introduction resulting from the experimenter's interpretation of the results, a post experiment debriefing was conducted in the presence of the three experimenters. While two of the experimenters were directly involved in proposing the principles, building the formal specification, and designing the experiment, a third one joined the team later to help in conducting the experiments.

5.5.5 Experimental Setup

The experiment was performed in the Software Engineering Research Laboratory at MIT using a DELL dual-CPU 2.8MHz Pentium XEON workstation with two 19-inch flat panel monitors. A single monitor was used when a single tool was available to answer the question. Both monitors were used (one for each tool) when both SpecTRM-RL and the visualization tool were available to answer the question. The room in which the experiment was conducted was closed and isolated from outside disturbances.

Chapter 6

Experiment Results

This chapter summarizes the results obtained during the experiment. The results and the observations made during the experiment are discussed in the light of our working hypotheses and of previous related work. The most significant results are mentioned in this chapter, but the complete quantitative results are provided in Appendix C.

6.1 Grading System

The result analysis includes two types of results: objective results based on the subjects performance, and subjective results based on observations and subject's perception of the experiment difficulty. Questions were graded on a 0-10 basis, based on a previously defined grading scheme. The number of points allocated to each question part is shown on the questionnaire provided in Appendix B. For questions with multiple answers, a negative grading scheme was used where points are awarded for correct answers and deduced for wrong answers. The grading was done at the end of the experiment, on the twelve questionnaires at once, in order to ensure consistency and fairness.

This grading system is by no means ideal. For experiment purposes, it would be interesting for the grader to know whether subjects made careless errors or whether they did not understand the specification. However, it can be argued that when reviewing safety-critical systems, errors should be avoided at all cost, regardless of the error context. Furthermore, trying to account for the subject's intentions could add unnecessary bias to the results. Consequently, we decided to use a grading system that does not take into account the error context.

The time spent answering each question was carefully recorded by the experimenter. A limit of 20 minutes per question was suggested but not strictly enforced by the experimenter. When the 20 minute period was over, the subjects were told that they should take a minute or two to finish what they were doing, and then move on to the next question. However, it

happened that some subjects refused to move on when they believed they were close to an answer. One subject in particular took over 50 minutes to answer the first two question parts. The subjects were told that the time spent on questions would be recorded, but that no points would be awarded for answering the questions quickly. This was done in order to reduce the stress that could result from adding a time component to the grade. Since the subjects were instructed not to rush through the experiment, time results should be interpreted in the context of the question results, rather than as stand-alone results.

After each question part, subjects were asked to subjectively evaluate the difficulty level of the question. Subjects had to rate each question part on a difficulty scale ranging from 0 (easy) to 10 (difficult). These results were used to assess the difficulty of each question and to identify behavior patterns.

In addition to these numerical results, the experimenters recorded comments from the subjects and information about problem-solving strategies and tool usage.

6.2 General Results

In many experiments with human subjects, inherent differences in ability between subjects create an important nuisance factor. The effects of this nuisance factor are even more important when a relatively small number of subjects participate in the experiment. When compiling the performance results of the twelve subjects, the first and easiest observation is the large variation in performance between subjects. Appendix C provides the complete quantitative results of the experiment. Figure 6-1 summarizes the answer accuracy of the twelve subjects on a 0-10 performance scale. When looking more closely at the average performance for each subject, it appears that the large variation comes from only two erratic data points: Subject #7 answered every single question perfectly, rapidly, and seemingly without effort, regardless of the tool used. On the other hand, subject #9 was struggling throughout the experiment, could not answer the questions without help, and grew discouraged and frustrated during the experiment until a point where he decided to leave without finishing the last question. This created some problems when compiling the results because the last question was missing a data point. However, it was predicted that some of these problems could occur as a result of the duration and difficulty level of the experiment.

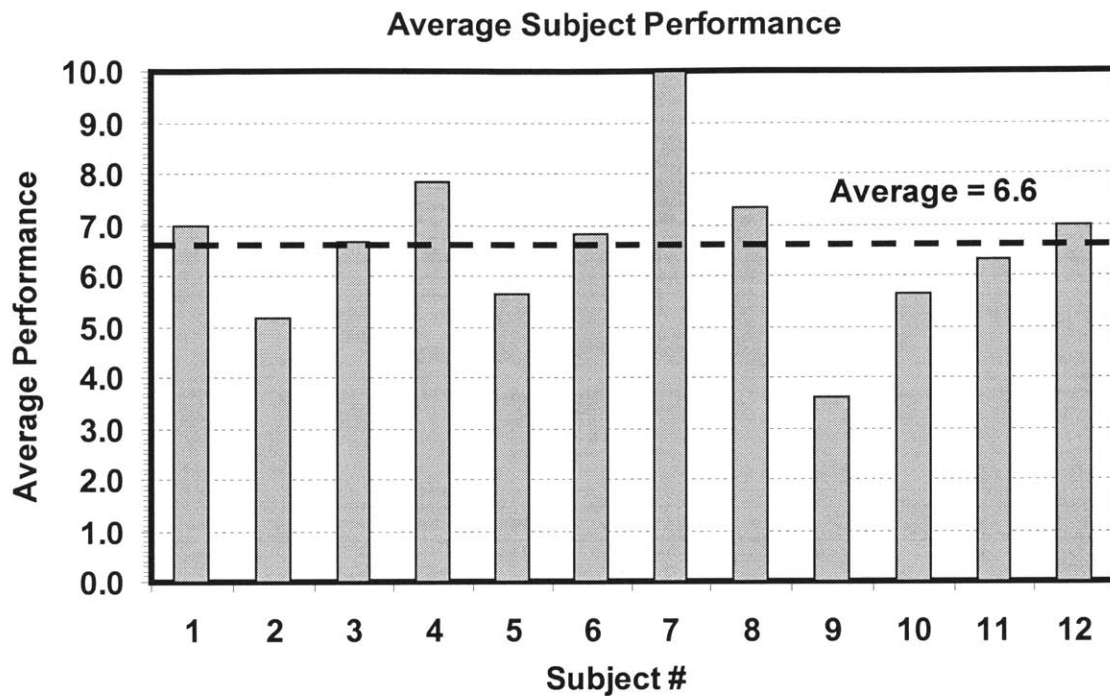


Figure 6-1: Summary of the average performance (answer accuracy) of each subject graded on a 0-10 performance scale.

Most of the variation in performance comes from subjects #7 and #9. Regardless of the tool used, subject #7 performed equally well on every question, while subject #9 performed equally poorly on every question. Because of this, the results of subjects #7 and #9 do not bring any insight to the experiment except for increasing the variation between results. Consequently, it was decided that the statistical analysis performed to confirm our hypothesis would not include subjects #7 and #9. We can justify this decision statistically by calculating the resulting standard deviation over the average results. The performance of subjects #7 and #9 is well outside the +/- 2 standard deviation envelope. Thus, those two subjects can reasonably be considered as outliers and omitted from the statistical analysis.

The results for the average difficulty rating and the average time spent per question for each subject are shown in figures 6-2 and 6-3, respectively. It can be observed that the outlying performance of subject #7 and #9 has a very strong correlation to the average time they spent on each task and on their perception of the task difficulty. Subject # 7 spent ten minutes per task on

average with an average question difficulty of 3.2, while subject #9 spent more than 20 minutes per question on average with an average question difficulty of 8. Such a strong correlation was expected and supports our decision to consider these two subjects as outliers.

Running a correlation analysis on the rest of the results exhibit a relatively weak negative correlation between the performance and the difficulty rating, as well as between the performance and the average time required to answer the questions. This is to be expected since most subjects demonstrate high consistency and confidence in their results. However, two subjects in particular demonstrate the opposite behavior in that a strong positive correlation is obtained between the performance and the time to answer and difficulty rating. This would suggest that a few subjects either underestimated the question difficulty and did not answer the question correctly, or had to work really hard to obtain the right answers. The subjects that demonstrate positive correlations performed under average.

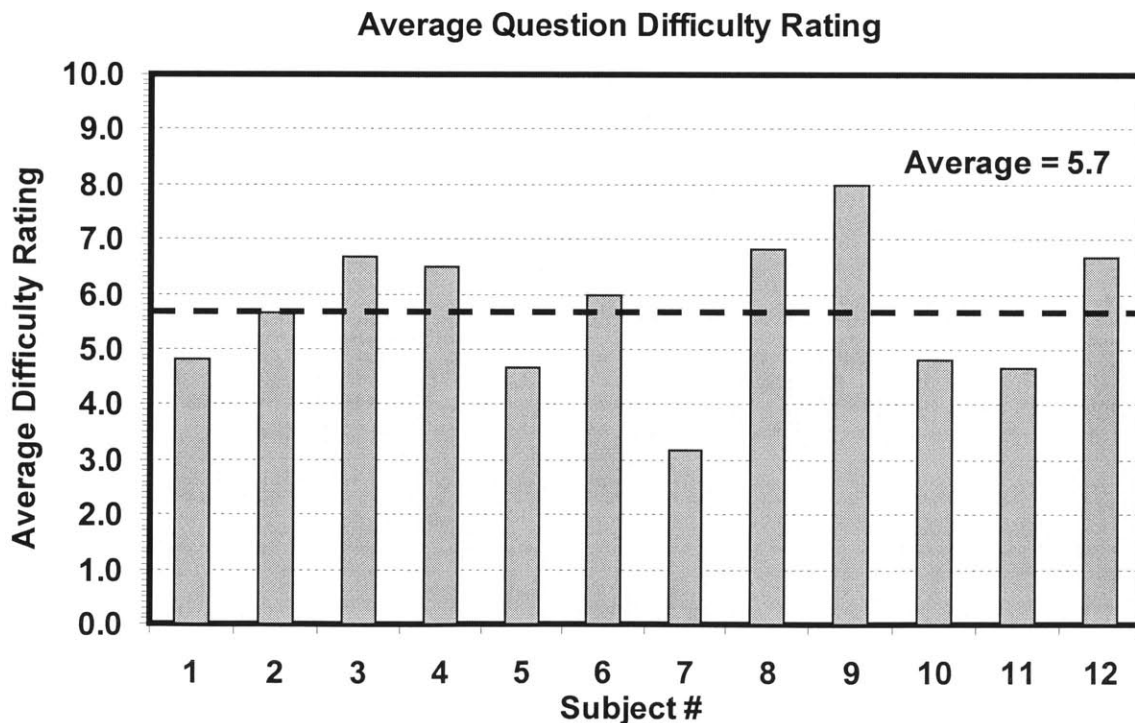


Figure 6-2: Summary of the average difficulty rating of each subject evaluated on a 0(easy)-10(difficult) scale.

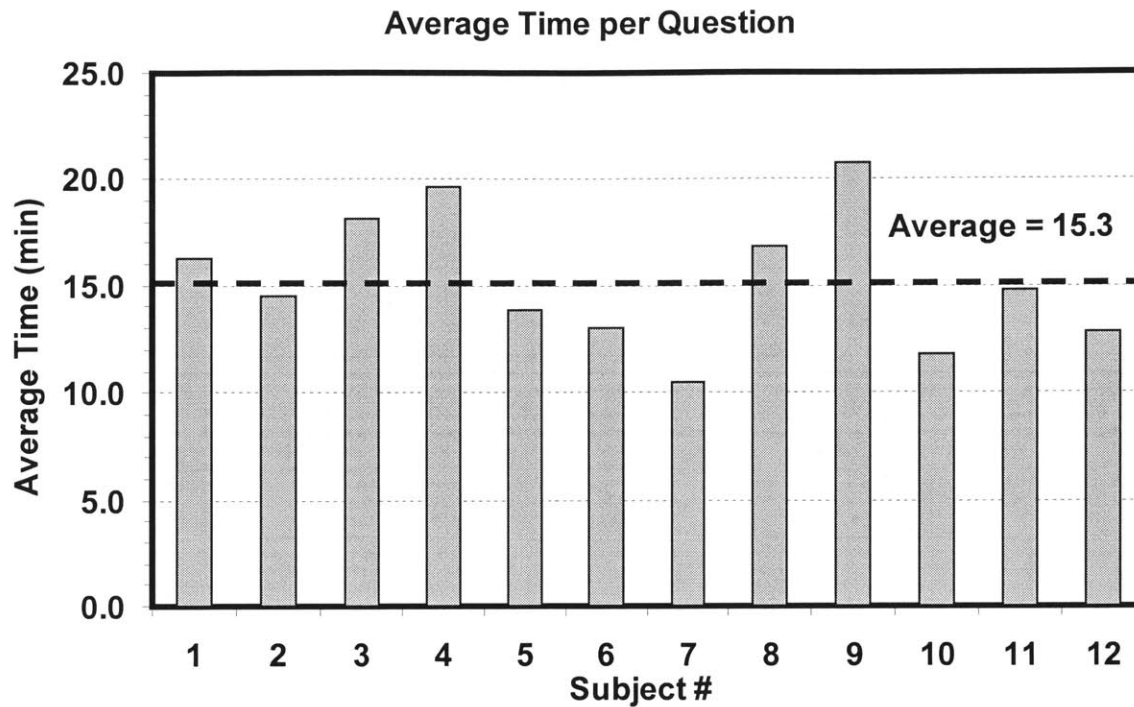


Figure 6-3: Summary of the average time spent answering each question.

6.3 Performance Metrics Results

This section interprets the results obtained in the context of the experiment hypothesis and performance metrics previously defined. A summary of the experiment performance metrics was provided in table 5-2.

6.3.1 Answer Accuracy Results

Figure 6-4 presents the overall average answer accuracy of subject using different tools. On a scale of ten, the average performance of SpecTRM-RL was 5.7, while the average performance of the visualization tool was slightly better at 6.5. More surprising is the fact that subjects having the opportunity to use both SpecTRM-RL and the visualization tool scored much higher at 7.3.

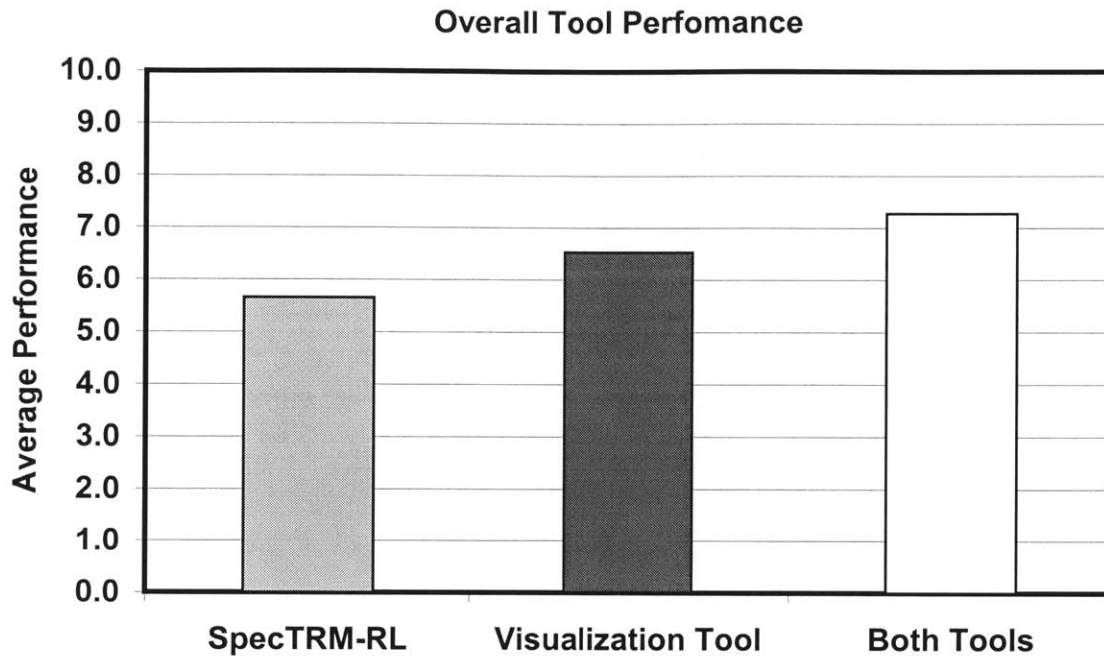


Figure 6-4: Average overall performance for subjects using the three different tool configurations.

Figure 6-5 shows the average answer accuracy for each question using the three different tool configurations. It can be observed that for most questions, the answer accuracy of subjects using the visualization tool alone compares well to that of subjects using SpecTRM-RL alone. The poor results of the visualization tool in the second part of Q1 can be explained in part by layout difficulties resulting from sub-optimal algorithms used while programming the tool. Q1-P2 is a behavior-oriented question that requires the display of the most complex state variable in the model. This state variable is specified using six pages of AND/OR tables in SpecTRM. Because of the highly complex behavior specified, the decision-tree is only readable when extended to a whole page, and when behavioral slicing is used. The resulting decision tree can be seen in figure E. A major inconvenience associated with this tree layout is that it requires more than 30 second to refresh itself. The interesting result is that subjects answering this question with both tools performed much better than subjects using the visualization tool only. Interestingly, when both tools were available, three subjects out of four used both at the same time, using the decision-tree to answer the question, and double-checking their answers using the AND/OR tables while the tree layout was refreshing. In this case, the slow refreshing time of the

visualization tool was a good motivation for users to use SpecTRM-RL tables. This could explain the high level of success obtained by users having both tools available.

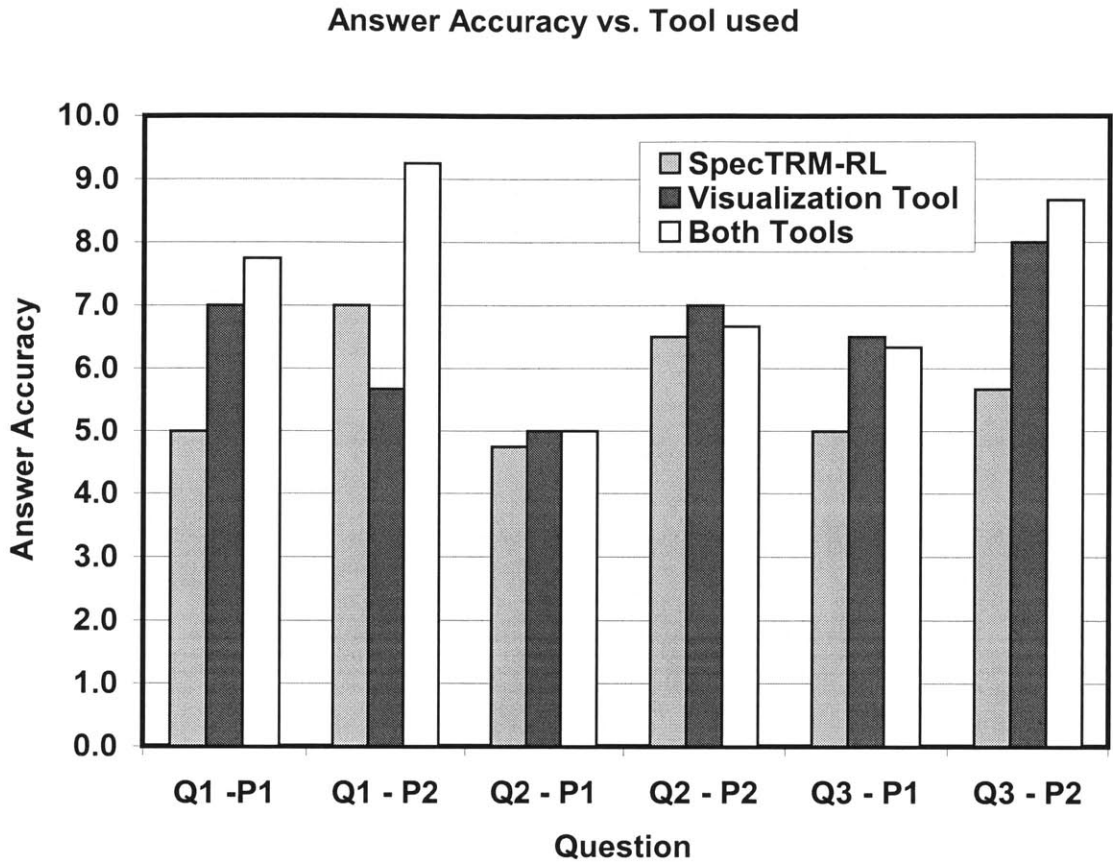


Figure 6-5: Average results obtained for each question part using the three possible tool configurations.

A paired sample two-tailed T-test was performed on the results in order to assess the statistical significance of the better performance of the visualization tool. Such a test is well suited for experiments with a few human subjects where a rather large variation in subject ability is expected. While the results show that subjects performed slightly better using the visualization tool alone than using SpecTRM-RL alone, absolutely no statistical evidence was obtained to support such a statement. However, when the performance of subjects using SpecTRM alone is compared to that of subjects using both tools, the test results reveal a probability of 97.5% that the performance difference is statistically significant. By most human experiment standards, such a result is convincing enough to state that statistically significant results have been obtained. Thus, it can be declared that a statistically significant increase in

performance resulted from answering questions with both tools available, as opposed to with SpecTRM alone.

On the other hand, the use of both tools also produced a noteworthy increase in performance over the visualization tool alone, even if a statistical analysis could not show with a high probability that such an increase is significant.

It does not appear that the background of the users had a large effect on the measured performance. Students with an aerospace background did slightly better, with an average of 6.7, compared to 6.3 for students with an EECS background. A statistical analysis did not reveal any significant difference in performance. Aerospace students performed better with SpecTRM-RL and AND/OR tables, with an average of 6.6, as opposed to 4.3 for EECS students. This would suggest that aerospace engineers find tables easier to use than computer scientists. Although we did not obtain strong statistical evidence to back up this claim, Zimmerman did obtain statistically significant similar results in his experiment on the readability of different transition conditions representations [40]. Using a similar two-tail T-test, Zimmerman was able to show at a 97.5% confidence level that when using AND/OR tables, aerospace engineers outperformed computer scientists. This result is slightly counter-intuitive since it could be expected that EECS students would do better at using SpecTRM because of their knowledge of discrete mathematics.

On the other hand, EECS students performed better with the visualization tool with an average of 6.8, compared to 5.8 for aerospace students. This could be partially explained by their better knowledge of graph theory and search algorithms. In fact, most EECS students seemed more at ease with the structural and behavioral slicing functions available in the visualization tool. The performance difference as a function of the subject background should be interpreted with care because no statistically significant results were obtained and because of the small number of data points available, especially when it comes to the performance of the EECS students, who displayed an extraordinary amount of variation in performance.

Another interesting observation comes from the scope of each question. Most questions were designed as hybrid behavioral/structural questions intended to test the subject's ability to integrate the information contained in the specification. However, a few questions were specifically designed as pure behavioral or structural questions. For instance, Q1-P2 is a pure behavioral question, while Q2-P2 and Q3-P2 are pure structural questions.

A possible explanation for the poor performance of subjects using the visualization tool for the behavioral question Q1-P2 was provided earlier. However, it should be noted that although subjects performed poorly using the visualization tool when compared to AND/OR tables, subjects using both tools performed incredibly well with an average score of 9.3. It can be argued from this performance that AND/OR tables and decision-tree representations of transition conditions are both valuable and complement each other very well.

When it comes to structure-oriented questions, every tool performed about equally well in Q2-P2. In fact, most of the errors made by subjects using SpecTRM had to do with a difficulty in remembering the dependency structure. Many subjects using SpecTRM had to take notes on paper and expressed some concerns about forgetting some links. On the other hand, most of the errors made using the visualization tool had to do with either a misuse of the structural slicing tool, or with a failure to consider multi-level dependencies. Some EECS subjects such as #1 and #7 knew immediately how to use the structural slicing function and were completely at ease with graph representations. Those subjects obtained a perfect 10 in both Q2-P2 and Q3-P2. Most subjects, however, were slightly uneasy with the slicing function at first, either using it completely wrong or failing to see multi-level relationships. It appeared, however, that most subjects got much better at using slicing as the experiment progressed. This learning process may explain the high level of success obtained by the visualization tool in Q3-P2. Moreover, every single subject having both tools available for Q3-P2 did not even consider using SpecTRM-RL to answer this question. However, one could argue that this type of purely structural question is a perfect match for the structural slicing function of the visualization tool. The lack of a structural overview in SpecTRM-RL makes it more difficult to answer this type of question.

6.3.2 Answering Time Results

Figure 6-6 presents the average time results for each tool/question combinations. As mentioned previously, the time results should be interpreted with care for many reasons. First, there was some saturation in the time results around 21 minutes because of the lightly enforced “moving on” policy. Second, some subjects used the extra time available to double-check their results while other subjects would just move on to the next question. For example, subject #4

was very careful to double check his results and made full use of the 20 minutes available for each question, completing the questions in exactly two hours. On the other hand, subject #10 raced through the experiment, completing the questions in an hour and ten minutes. Finally, since the subjects were told that answering the questions quickly would not increase their performance, the time results obtained should be used for insight purposes only.

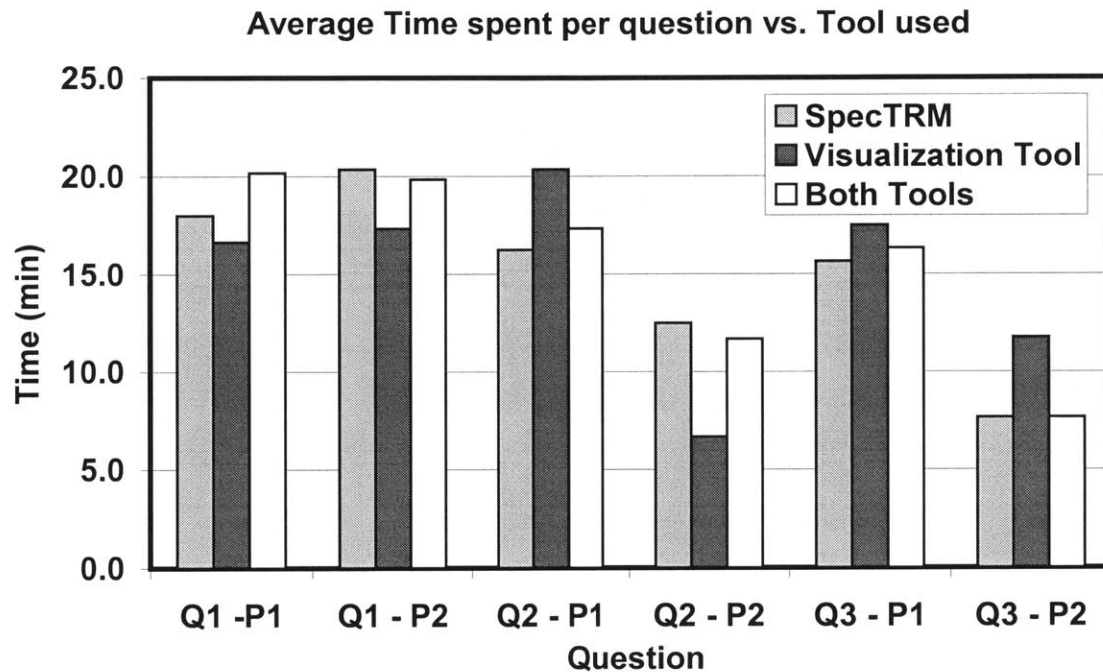


Figure 6-6: Average time spent on each question part using the three possible tool configurations.

Statistical analysis does not reveal any significant difference in the time required to answer the question as a function of the different tool used. The very quick answer to Q2-P2 by the visualization tool users is a result of a subject underestimating the question difficulty and failing to identify indirect relationships between elements, thus obtaining completely wrong answers. In general, the shorter time required to answer Q2-P2 and Q3-P2 is a consequence of the structure-oriented questions being somewhat easier and more straightforward than the others.

Little insight was gained through the analysis of the time required to answer the questions when it comes to the different tools used. However, when comparing the results of the subjects with different backgrounds, some interesting results emerge. Indeed, it was possible to show a

statistically significant difference in average time required to answer questions between students with an aerospace background and students with an EECS background. On average, EECS subjects took 82 minutes to answer the questions while aerospace subjects took 97 minutes. A two-tailed T-test analysis based on these results shows at a 99% confidence level that on average, EECS subjects answered the questions faster than aerospace subjects. This supports Zimmerman’s observation that computer science students completed his experiment significantly faster than aerospace students [40].

6.3.3 Question Difficulty Results

Figure 6-7 presents the results of the overall average difficulty ratings for each tool. With an average difficulty rating of 5.6, the visualization tool seemed to slightly decrease the apparent question difficulty when compared to SpecTRM-RL (6.3) alone. However, when both SpecTRM-RL and the visualization tool were available, users evaluated the difficulty rating to be even lower (5.3). This suggests that in addition to providing better performance, a combination of the tools could decrease the apparent difficulty of the tasks.

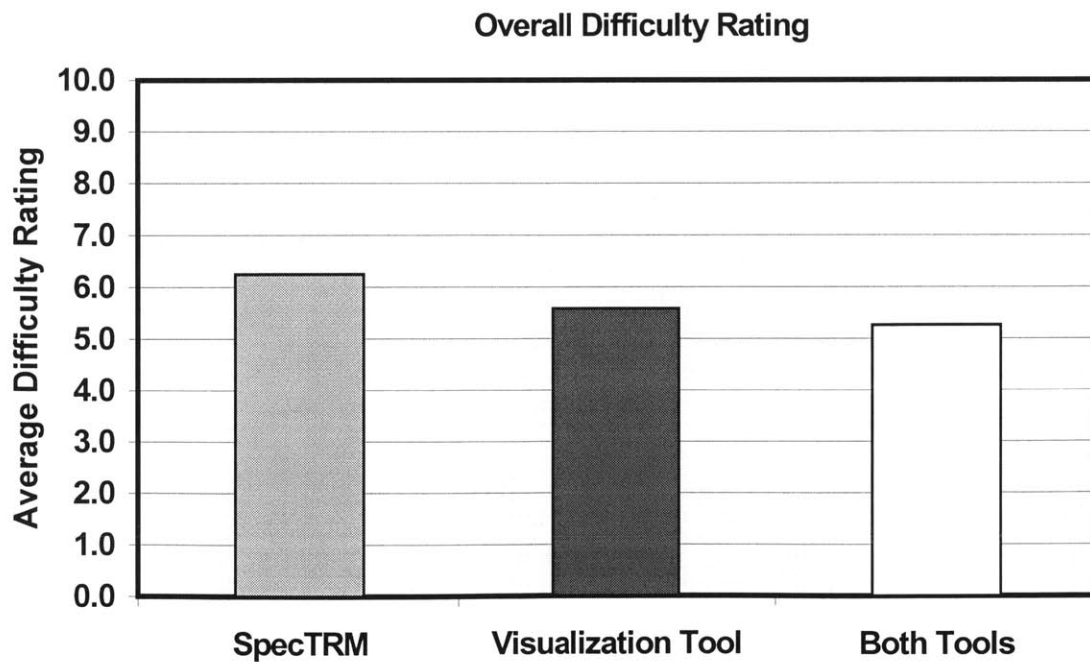


Figure 6-7: Average overall difficulty rating for subjects using the three different tool configurations.

Figure 6-8 presents the average difficulty rating for each question as a function of the tool used. Although the results suggest that, on average, users had more difficulty answering questions using SpecTRM-RL, no strong correlation was found to exist between the performances achieved using different tools and the associated difficulty rating. In fact, the visualization tool seems to create an illusion of facility that may result in subjects underestimating the question difficulty and making careless errors. As an example of this, subjects # 10 and #11 did very poorly on Q2-P2 using the visualization tool. In both cases, the users completely underestimated the question difficulty, resulting in a failure to notice the indirect relationships between elements that led to completely wrong answers. Although subjects using SpecTRM-RL alone had to work harder to obtain an answer in Q2-P2, the organization of the information in SpecTRM-RL enforced the detection of the indirect relationships that subject #10 and #11 overlooked.

A very strong positive correlation was observed between the average difficulty rating and the time required to perform the tasks. This was expected since users having some difficulty answering a question would most likely spend more time working on it.

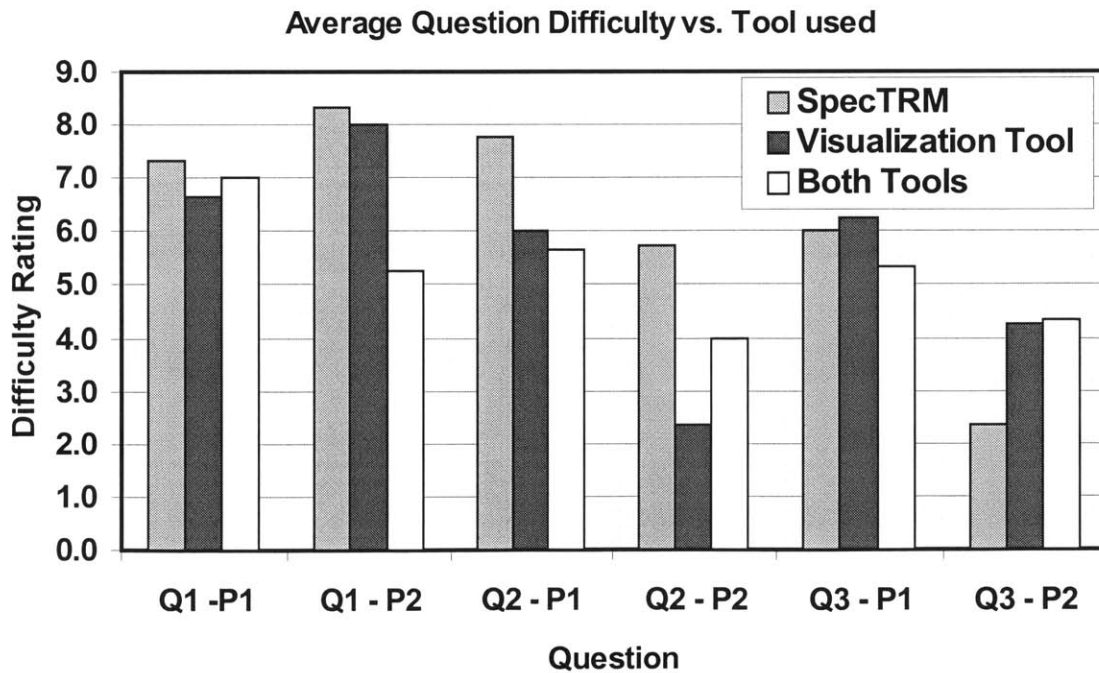


Figure 6-8: Average difficulty rating for each question part using the three possible tool configurations.

6.4 User Preferences

Another reason why we decided to provide both tools to answer some questions was to observe which tool would be chosen, thus trying to confirm the unstated hypothesis that subjects will be instinctively attracted to rich visual representations. This was confirmed during the experiment as every single subject used the visualization tool when available. Out of the twelve subjects, ten went immediately for the visualization tool, while two decided to use both tools at the same time. Two subjects used only the visualization tool to answer the questions, while the others used a combination of the two. In general, subjects had a tendency to use the visualization at first, while using SpecTRM-RL to perform backup tasks such as text searches. Many subjects mentioned that the lack of a text search function is the most important disadvantage of the visualization tool. Neither subject used SpecTRM-RL alone when the visualization tool was available. This supports Petre's observations that subjects are instinctively attracted to colorful, visually attractive representations.

Interestingly, one of the computer science subjects started answering behavioral questions using AND/OR tables because of his computer science background (his words), however, when confronted with the large tables of Q1-P2, the subject got worried and changed strategy saying: "I should use the tree, it'll be easier... When the tables get bigger, it seems easier to use the tree." However, after using the decision-tree for a while the subject got confused with the behavioral slicing tool and had to check his answers using the tables, which he did very well. Most subjects liked the decision-tree better over the AND/OR tables because "the tables have no memory" as one subject put it. Users had the possibility to use interactive behavioral slicing with the decision-tree, which removed the need to remember which conditions have already been evaluated. Although it is possible to simplify the AND/OR tables based on an operating scenario in SpecTRM-RL, the subjects were not taught the technique because it was decided that the tutorial would be too long. This can be seen as an unfair advantage that may have biased the user's preferences toward the visualization tool.

When asked to explain why the visualization tool was preferred over SpecTRM-RL, subjects had comments such as: "The visualization makes it easier to tie things together... It's tough to see the big picture in SpecTRM-RL, too much to remember...", "It's easier to see what's going on with the visualization, there's too much clicking back and forth in SpecTRM-RL", and "The visualization is easy to use except for programming bugs, it helps to see the

relations because everything is on one page.” In general, the users seemed attracted to the interactivity of the visualization tool, which gives an impression of control over the specifications, even though the information remains unchanged. However, the powerful interactive features of SpecTRM-RL (slicing, analysis, model execution, etc...) were not available, which may have affected the user’s preferences.

6.5 Principle-Specific Discussion

In the preceding chapter, four principles were identified as the most interesting to evaluate in the context of this experiment. In this section, the application and validity of these principles is discussed in the light of the observations made during the experiment.

6.5.1 Highlight Hidden Dependencies

The specification used for this experiment included two types of hidden structural dependencies. The first type of hidden dependency is a side effect of the high level of coupling between elements of the model. The value of many state variables depends on a large amount of information coming from different elements which themselves affect the value of many other elements. This type of hidden dependency occurs when individual dependencies are obscured or buried within the mass amount of information. This occurs as much in SpecTRM-RL as in the unsliced structural overview (V1). Structural slicing seemed to be the preferred way for subjects to cope with this type of hidden dependency. The use of structural slicing in V1 appeared to be fairly intuitive to users having a prior understanding of directed graphs and some idea of the meaning of slicing. However, some users had to learn the meaning of slicing the structural overview. Those users improved throughout the experiment but were not as proficient as the users who instinctively knew how to use the tool. One subject in particular misused structural slicing because of a complete lack of understanding of its meaning. Most subjects were able to isolate structurally coherent subsets, which was useful in several questions. They were also able to easily transition from sliced to unsliced view. It is strongly believed that the fact that context

remains visible to ensure geometric continuity during slicing is the main factor explaining this apparent ease-of-use. However, we do not have objective evidence to backup that claim.

A major shortcoming of V1 is that it does not allow input and output based slicing. In order to isolate such structural patterns, subjects had to manually search the “raw” data since V1 was not able to highlight these dependencies. On the contrary, when compared to SpecTRM-RL, V1 tends to obscure these dependencies. Most subjects were confused by what appeared to be an inconsistency within V1, and performed poorly on the questions that involved input influence. It is believed that the additions of a simple input and output based slicing feature could eliminate this problem.

The second type of hidden dependency involves the indirect structural dependencies (structural dependencies involving multiple levels of dependency) naturally “hidden” in both SpecTRM-RL and the unsliced V1. Indirect dependencies exist because explicitly representing each dependency for the entire model would be impractical and would add unnecessary size and complexity. Indirect dependencies are an important aspect of any specification because they are involved in all questions of the kind: “What if that component fails?” or “What happens to the system if I make a change here?” V1 offers a simple and efficient way to visualize these dependencies by applying multiple levels of slicing. Subjects who understood the concept of indirect dependencies were able to answer this type of question much faster and with greater accuracy using V1 because SpecTRM-RL does not provide an explicit representation of indirect dependencies. On the other hand, subjects who were not comfortable with the concept of indirect dependencies had a tendency to perform better using SpecTRM-RL because the lack of a structural overview enforces the link-following process that inevitably uncovers indirect dependencies. In summary, using SpecTRM-RL for purely structural questions required more effort, but was somewhat safer.

Our observations show that this principle is extremely valuable. It is important because exploring the dependency structure of a specification is a tricky, yet frequent and important review task. In the case of hidden structural dependencies, nodes-and-links graphs appeared to be a natural and more effective representation in terms of the amount of effort necessary to obtain a solution, but it required more training than the hyperlink representation used in

SpecTRM-RL. The task of the visualization designer is to identify all possible relevant types of hidden or indirect dependencies and find ways to properly highlight them.

6.5.2 Support Top-Down Review

This principle was less controversial than the others. A great deal of evidence already demonstrated that top-down review is an effective reviewing approach [21, 24]. For that reason, the top-down approach was tightly incorporated, and even somewhat enforced in the visualization tool. In fact, it is impossible to access the detailed information in V2 without first locating the desired element within the structural overview (V1). The experiment shows that users have absolutely no problem with that approach. Users did not seem to be upset by the necessity to navigate the overview in order to access detailed information. These observations confirm that top-down review is a solid, proven approach to deal with very large amounts of information.

Another part of the top-down review principles states that the use of an overview promotes the *gestalt* effect, which allows the emergence of the overall structure and promotes the construction of a mental model of the system. One subject clearly demonstrated the value of V1 for quickly creating a high-quality mental model of the system by finding an inconsistency in the specification solely based on the structural overview (V1). During the tutorial, subjects were presented with a high-level hierarchical overview of the VG Annunciation Process and its interfaces. This high level overview is shown in Figure F. The tutorial clearly explains that the Display Electronic Unit (DEU) device is responsible for processing the outputs to the Primary Flight Display (PFD) and Navigation Display (ND). Question Q2-P1 asks: “The FMA speed window is used to display speed targets on the pilot’s primary display unit (PFD). What additional information is required for this window to display a magenta speed target?” The subjects were expected to first locate the required state variable *FMA Speed Magenta-White Discrete*, and then use the information available in the information panel to ensure that it was the right element to answer the question. However, one subject pointed out that it makes no sense that FMA Speed Magenta-White Discrete provides outputs only to the FCP device since the question mentions that this state variable controls the display on the PFD, which is not connected

with the FCP. After verification, it turned out that the subject was correct and that an external link was missing between the FCP and the DEU, which controls the PFD. The model in itself was correct, but some inconsistency resulted from the omission of the external link added as a dotted line in figure 5-4. The user was able to detect an inconsistency between the information in the question and his mental model of the system created through the use of the structural overview (V1).

While the visualization tool provides a graphical overview of the structure of a model, SpecTRM-RL provides a graphical overview of the model behavior (see figure 2-3). The behavioral overview displays the inputs and outputs to the system, the modes and state variables of the system, along with their possible values. During the periodic reviews by domain experts of a formal specification of a collision avoidance provided for the FAA, Leveson observed that such a graphical behavioral overview of the entire specification was a very powerful review tool [7].

Although such a behavioral overview was available in SpecTRM-RL for the *VG Annunciation Process* system, the automatically generated layout of the graphical overview was very awkward because of the sheer size of the model used. Consequently, subjects did not use the graphical behavioral overview. The commercial release of SpecTRM-RL features greatly improved graphical overview layout generation, but it was not used in the experiment because it was not available at the time.

The characteristics of the overview will support different types of review tasks, some requiring more expertise than others. For example, a graphical overview of the system behavior such as that provided in SpecTRM-RL may be very useful for reviewers having a refined mental model of the system's functioning, while a structural overview may be more useful for the initial creation of the mental model.

Nevertheless, the use of hierarchy is the preferred way for humans to manage complex systems. Consequently, top-down review is an extremely important, well-accepted principle that should be supported by visualizations designed to enhance the reviewability of formal specifications. The fact that a subject was able to detect an inconsistency in the model based solely on the structural overview is a proof of the value of the principle.

6.5.3 Support Alternative Strategies

The flexibility of the visualization tool was artificially limited because the experiment subjects were all novices. Implementing the visualizations with a large number of features increases their potential and flexibility, but makes them more difficult to use without extensive training. Since we wanted the pre-experiment tutorial to be limited to an hour, some of the features were suppressed, such as the ability to visualize several graphs at the same time, or the ability to reorder columns in the decision-trees. Some features usually available in SpecTRM-RL were also restricted because they would have required too much training to master. These restricted features included slicing, analysis and model execution.

However, some flexibility was preserved and several questions were formulated in such a way that users could obtain an answer using many different strategies. For instance, V1 does not provide any preferred “entry point” to the system: a search strategy can start with inputs, outputs, or any internal element. Most of the time, an answer to the questions can be reached through several approaches. Although it is believed that some of the most powerful features cannot be used without some level of expertise, some simple features such as text searches and input/output based slicing were often mentioned by users as being most required for the visualization tool. The addition of these features probably would have added valuable flexibility to the visualization tool without the need for much additional training.

Most subjects displayed a very high level of flexibility in problem-solving strategies. This is consistent with Vicente’s observations [37] that:

“[...] the detailed cognitive procedures used by workers during any one particular situation are idiosyncratic. As a result, there is a great deal of variability across situations. [...] This variety is also due to individual differences in behavior, both across workers within a situation and within a worker across situations. Different people prefer to perform the same tasks in different ways. Moreover, even the same person can prefer to perform the same task in different ways on different occasions.”

Because of this, problem-solving strategies are usually defined as a category, rather than as an explicit sequence of operations. Rasmussen proposed a definition of strategy in this context: *“A strategy is a category of cognitive task procedures that transforms an initial state of knowledge into a final state of knowledge.”*

The use of a single tool to answer the questions somewhat limited the number of different strategies subjects could use. Despite a large variability in individual problem-solving strategies, most users seemed to reach answers in a very systematic way. They would start answering questions using the strategy that instinctively seemed more appropriate. When both tools were available, this strategy usually included the use of the visualization tool. However, as soon as a roadblock was reached, instead of pushing the use of the initial strategy, users having another tool available had a strong tendency to switch tool in order to overcome the difficulty. As an example, such change in strategy was very often encountered when users could not find the required elements in the visualization tool. If SpecTRM was available, they would locate the desired information in SpecTRM, take note of some additional cues in order to locate the information in the visualization tool and continue on their initial problem-solving path. This is consistent with Rasmussen's observations that individuals make *spur-of-the-moment shifts in processing directions* rather than merely executing some strategy determined beforehand.

Also, when users had limited confidence in an answer reached, they would very often double-check this answer using the other available tool. In some cases, it appeared that this double-checking process revealed an easier way to answer the question and the users would re-answer the question using the easier strategy.

In general, subjects would obtain a solution using a strategy similar to a depth-first search strategy with backtracking on failure. The increase in performance resulting from the availability of both tools could be partially explained by the smaller backtracking depth that could be achieved by using the other available tool. As soon as a difficulty arose, users would seek alternate answer paths using the other available tool, instead of persevering in an uncertain direction that would often lead to a dead-end. Surprisingly, subjects who employed many strategies as a result of the failure of their first strategy performed better than the others

This category of strategy was identified by Rasmussen and Jensen [27,28] as "The Technician's Approach". While studying the diagnosis of equipment failures, they observed that rather than exploit knowledge of the equipment to develop an equipment-specific plan of attack, professional technicians tended to use generic methods that were independent of both the equipment and the fault. Rather than extracting as much information as possible from each observation, technicians tended to use each observation only to determine where to make the next observation. Various strategies have different resource time, memory, and knowledge

requirements. Rasmussen and Jensen found that when one strategy is becoming too effortful, technicians would spontaneously switch to another strategy to meet the task demands in a more economic fashion.

It is interesting to note that although every subject had an undergraduate degree in engineering or computer science, the strategies observed were much closer to the “Technician’s Approach” than to the “Engineer’s Approach”, which is based on a profound knowledge of the system’s functioning. It appears that given the time constraints and the impossibility for subjects to obtain a deep understanding of the system within a two-hours period, subjects performed the experiment tasks as professional technicians, jumping from one part of the specification to another, collecting bits of information on the way, in order to decide where to go look next. Given the constraints, subjects automatically switched to the “Technician’s Approach”, which was facilitated by the larger flexibility resulting from the availability of both tools. The strategies observed confirm this hypothesis and at least a portion of the performance increase can be attributed to the larger flexibility in strategies offered to subjects having both tools available.

6.5.4 Provide Redundant Encoding

It was believed that providing both the visualization tool and SpecTRM at the same time could enhance the users’ performance by providing them with the opportunity to choose the most appropriate representation for their tasks. However, it seems that in most cases, users decided which tool would be used *before* reading the question. This choice was based on subjective criteria, rather than on analysis of the task to be performed. This can be explained by the inexperience of subjects who could not identify beforehand the most effective strategy to answer each question. Users had a tendency to start with one preferred notation and switch only if this notation failed. However, further experimentation using possibly more experienced subjects would have to be performed to see if users would eventually choose one representation over another based on more systematic criteria.

Although it appears that subjects did not choose a particular representation based on objective criteria, the background of subjects using different representations seemed to affect their performance. The results mentioned previously seem to demonstrate that aerospace subjects are better at using SpecTRM-RL and AND/OR tables, while EECS students are better at

using the visualization tool. This supports the results obtained in a previous experiment on the readability of various notations where Zimmerman was able to show at a 97.5% confidence level that aerospace subjects find tables easier to use than computer science subjects. These combined results suggest that multiple representations of the same information may support reviewers with different backgrounds. In this case, an emerging problem will be to find a way to encourage reviewers with different backgrounds to choose a representation that helps to optimize their performance.

Every representation exhibits different strengths and weaknesses. When both SpecTRM-RL and the visualization tool were available, many users would somehow create a more powerful “hybrid” representation by manually compensating for the weakness of a representation by using the other. This supports both the need for more flexibility and the need for redundant encoding of the information contained in the specification.

This experiment very clearly demonstrates the value of the redundant encoding principle. Whether it is because users with different background will prefer different notations, or because the characteristics of a representation will make it more suitable to perform a certain type of tasks over another, this experiment strongly corroborates the hypothesis that *multiple representations based on the same underlying formal model will improve the specification reviewing process.*

6.6 Experiment Limitations

Running the experiment on human subjects revealed limitations that were overlooked during the preparation of the experiment. Some of these limitations could potentially have adverse effects on the initial objectives of the experiment.

One of the limitations comes from the relatively small number of subjects who participated in the experiment. Based on previous experiments performed at SERL, it was decided to limit the number of participants to twelve, in order to prevent excessive experiment duration. Theoretically, twelve subjects are sufficient to obtain statistically significant results, however, a

small number of subjects decrease the robustness of the results obtained and the tolerance to outliers.

The user background may also be a limitation. It was decided to recruit graduate students at MIT rather than professional systems engineers because students were more available and provided a relatively homogeneous group of subjects that had little or no exposure to specification reviewing tasks. Because of this, some of the answers may be based on intuition rather than on a rigorous systems engineering process, which would have been the case if professionals had been used.

Another limitation was the possible bias introduced by the experience of some of the experimenters. In order to evaluate the visualizations created for reviewing formal specifications, questions were designed to reflect realistic tasks that professional reviewers perform on a routine basis. Although every care was taken to ensure that questions were representative of real tasks, it is possible that the question designer's relative inexperience with formal reviewing processes influenced the question design. Also, the question designer was directly involved in the design of the visualization tool. Independent people reviewed the questions but some bias may have been unintentionally introduced by the question designer's familiarity with the features of both SpecTRM-RL and the visualization tool.

Another important limitation is the scope and duration of the experiment. In order to reflect real-world reviewing tasks, the questions were necessarily challenging. Although such questions allowed us to thoroughly test the subjects' ability to use the different tools provided, some of the subjects quickly grew tired and discouraged. This may have affected the answers to the last questions. One of the objectives of the experiment was to observe whether the subjects were able to dynamically adapt their problem-solving strategies. This requires some time for the subjects to learn which strategies work in certain situations and to adapt. Consequently, from an experimenter's point-of-view, even longer experiment duration would have been desirable. However, from a subject point-of-view, it was simply unreasonable to extend the total experiment duration beyond three hours.

Experiment subjects were all novices in the field of formal requirements specification. Consequently, an hour-long tutorial was necessary for the subjects to be able to understand the specification and to effectively use the tools provided. The tutorial combined with the nearly

two-hour experiment extended the total duration well beyond two hours for most subjects. However, it appeared that this tutorial was not sufficient for some of the subjects to acquire the necessary skills. Insufficient training for some subjects was reflected in their lack of confidence in answering some questions. It could be argued that such variation is simply due to natural ability differences between subjects, but it is believed that an overly intensive learning process accentuated those differences. In the real world, individuals usually have the opportunity to extend learning activities over a much longer period of time and have more time to practice.

6.7 Recommendations

Based on the experiment limitations and sources of error mentioned previously, we suggested several recommendations intended for further experimentation in this field.

People with various backgrounds use complex systems specifications. The reviewability of formal specifications by non-technical people is an important factor in the acceptance of formal methods by the industry. Consequently, it would be useful to test subjects with a non-technical background. Although every experiment subject possessed an undergraduate background in a technical field, it should be mentioned that the subject who answered every question perfectly is doing graduate work in a highly technical field while the subject who did poorly on the experiment has spent the last years pursuing graduate studies in a management/policy-oriented field. It is not possible to extrapolate from those two data points and claim that technical people are better reviewers than non-technical people, but further investigation could shed some light on the matter.

Providing more training on the use of state-machines or selecting subjects with a better understanding of the functioning of state-machines would also be useful. Although state-machines are a relatively easy concept that is quite natural for engineers, it is unreasonable to think that a ten-minute introduction to state-machines is enough for subjects to become proficient in the subject and confident in their answers. Furthermore, industry reviewers would certainly be familiar with the concept, or at least given enough time to understand it well.

Another recommendation is to include professional reviewers in a future experiment. Graduate students were used in this experiment for convenience and availability, but their lack of reviewing experience was a limitation when it came to associating particular tasks with effective problem-solving strategies. Experienced reviewers would bring more insight into this matter.

If this experiment had to be repeated, a stricter enforcement of the time limit should be considered. Although most subjects could answer the questions in less than 20 minutes, some subjects took significantly more time to answer some questions, which may bias the results.

The experiment was too demanding for most subjects. The combined tutorial and questions required a good amount of concentration from the subjects. Maintaining such a concentration level for almost three hours ended up being too much for some subjects, who grew tired and frustrated. Further experiments should be divided in two parts. Subjects should go through a longer tutorial session on a day, and answer the experiment questions the following day. Each session should be limited to two hours.

Chapter 7

Conclusion

Reviewability of formal specifications is one of the major roadblocks hindering industry adoption of formal methods. This research is founded on the working hypothesis that visualization can significantly improve the specification reviewing process, thus supporting the industrial acceptance of formal specifications. The experiment performed as the core of this thesis highlights strong trends that support this belief.

7.1 Attractiveness and Performance of Visual Representations

The most obvious trend is the attractiveness of rich visual representations. This phenomenon had already been thoroughly observed and documented [24]. When given a choice of representations, ten out of twelve subjects initially used the visualization tool over SpecTRM-RL, while two subjects initially used both representations at the same time. Although most users chose to initially use the visualization tool, as soon as they encountered a difficulty, most subjects were very inclined to use SpecTRM-RL to overcome the difficulty. Nevertheless, the natural appeal observed toward rich visual representations comforted our belief that visualization can actively participate in making formal methods acceptable to a larger community of engineers and researchers.

Petre [24] concluded that even if representations containing rich visual content do not improve the performance of users compared to more traditional representations, their attractiveness in itself may be their major strength and should not be underestimated. However, our results show that users performed well using visual representations. Although the performance of subjects using SpecTRM-RL alone was comparable to that of subjects using the visualization tool alone, a significant increase in performance resulted from the availability of both tools. It was even possible to show at a 97.5% confidence level that the combination of

SpecTRM-RL and the visualization tool resulted in a significant increase of performance over SpecTRM-RL alone. Similarly, the combination of both tools created an increase of performance over the visualization tool alone, but the significance of the statistical results was not as strong.

The main objective of this experiment was to evaluate the interactive visualization tool created based on the proposed design principles. SpecTRM-RL was used as a basis for comparison but it would be a mistake to equate the visualization tool to a pure visual representation and SpecTRM-RL to a pure textual representation. In fact, the decision-tree representation of transition conditions includes just as much text as the table representation used in SpecTRM-RL. However, it can be reasonably argued that the visualization tool takes more advantage of perceptual cues such as color and position than SpecTRM-RL does. Since the combination of SpecTRM-RL and the visualization tool resulted in better performance, it suggests that the best reviewing tools should include a combination of many different representations, each highlighting different properties of the information contained in the specification.

7.2 Validity of *Highlight Hidden Dependencies Principle*

The observations made during the experiment show that highlighting hidden dependencies is a very important design principle. Ideally, dependencies should be highlighted in such a way that detecting hidden dependencies is effortless and unequivocal. However, it appeared that while the visualization tool provided a simple and efficient way of highlighting hidden dependencies by applying multiple levels of slicing, some subjects were not able to take advantage of this feature because of a misunderstanding of its purpose. Those subjects performed better using SpecTRM-RL because the lack of a structural overview enforces the uncovering of indirect dependencies at the price of larger effort. In short, using SpecTRM-RL seemed to be a safer way to explore the structural dependencies of a model, but it usually required more time and effort. Also, the different properties of each representation seem to highlight different types of dependencies. For example, questions about the influence of input values seemed to be easier to answer using SpecTRM-RL because the visualization tool was not

able to highlight this type of dependency. In addition to supporting the “Highlight hidden dependencies” principle, these observations seem to show that different types of dependencies can be highlighted through the use of redundant encoding.

7.3 Validity of *Top-Down Review* Principle

Top-down review was somewhat enforced in the visualization tool because it was impossible to access detailed behavioral information without first locating the desired element in the structural overview. The subjects seemed to adopt very quickly the use of an overview as a navigation tool. Some comments even mentioned that the structural overview helped to tie things together and understand the relationships between elements. In particular, one subject was able to detect an inconsistency in the model based solely on the structural overview. The use of a graphical overview, either structural (visualization tool), or behavioral (SpecTRM-RL), seems to be a good way to promote the creation of a mental model of the system. These observations, along with past experiences confirm that top-down review is a solid, well-accepted approach to deal with large amounts of information.

7.4 Validity of *Support Alternative Strategies* Principle

The results show that the most important factor affecting performance was arguably the flexibility built in the representation and the possibility to answer questions in many different ways. The large variation in problem-solving strategy between users was quite amazing to observe. This large variation may be due in part to the inexperience of users who did not know beforehand which tool or strategy would be better suited to answer certain types of questions. However, such a large variation in strategy was expected based on the results of other experiments with human subjects. Rasmussen explains this by the idiosyncrasy of the detailed cognitive procedures used by different people in a particular situation. In addition to exhibiting different strategies, some subjects switched strategy many times while answering a single question. This is also consistent with Rasmussen’s observations that individuals make spur-of-the-moment shifts in processing directions rather than merely executing a strategy determined

beforehand. Interestingly, the subjects provided with more flexibility through the availability of multiple representations undoubtedly performed better. This supports our claim that much flexibility must be built in the visualizations in order to support users with different backgrounds and expertise levels in performing a large variety of reviewing tasks.

7.5 Validity of *Provide Redundant Encoding* Principle

As mentioned previously, some of the principles may overlap. As an example, redundant encoding both supports the highlighting of different types dependencies and provides much needed additional flexibility in problem-solving strategy. In addition to supporting these two principles, it appears that users with different backgrounds performed better using different information representations. For instance, it seems that aerospace subjects were better at using SpecTRM-RL than EECS subjects. This claim was not backed by strong statistical evidence and may be anecdotal but it supports previous results showing at a 97.5% confidence level that aerospace students were better than computer science students at using AND/OR tables. On the other hand, EECS students performed slightly better using the visualization tool than aerospace students, possibly because of their knowledge of graph theory and search algorithms. These results further support the redundant encoding principle by suggesting that multiple representations of the same information may increase the performance of reviewers with different backgrounds.

7.6 Other Results and Observations

It appeared that subjects improved at using the visualization tool throughout the experiment. Apart from a few exceptions, subjects were confused at first and had trouble understanding the meaning of some functions of the visualization tool. However, toward the end of the experiment, most subjects exhibited more confidence and ability while using the visualization tool. A linear

regression analysis based on the subject performance with different tools show a net positive improvement in performance using the visualization tool.

There was no indication that one tool permitted answering the questions faster than another. However, the time results may be biased by the different attitudes of subjects during the experiment. Some subjects were very patient and meticulous, paying attention to details and verifying their results while other subjects would move on to other questions very quickly, even when unsure about an answer. Although no significant time difference was observed between subjects using different tools, it was possible to show with a 99% statistically significant confidence level that with comparable performance, EECS subjects performed the experiment faster than aerospace subjects. This result confirms observations made in previous experiments.

The difficulty ratings collected seem to show that for the particular questions asked, users perceived the visualization tool to be slightly easier to use than SpecTRM-RL. However, when combining the difficulty ratings with the answer accuracy, results shows that subjects often underestimated the question difficulty when using the visualization tool, resulting in completely wrong answers. SpecTRM-RL users never underestimated the question difficulty. Although users often had to work harder to obtain an answer using SpecTRM-RL, it seemed to be less error-prone than the visualization tool. Interestingly, the results show that the combination of both tools, in addition to providing better performance, significantly decreased the apparent difficulty of the tasks. This result is encouraging because it suggests that the use of the design principles could not only increase the performance of specification reviewers, but also decrease the apparent difficulty of typical specification review tasks.

More visualization designs, along with more human experimentation will be required to build confidence in the design principles proposed in this thesis. However, the results obtained are encouraging and should be seen as a good motivation for further research in the area.

The results of this research show that visual representations of formal requirements specification have strong potential, especially as a complement to more traditional representations. It is believed that the development of easy-to-use, intuitive formal specifications tools including powerful features and analysis capability is the keystone to the industrial acceptance of formal specifications. The potential benefits justify the effort.

References

- [1] Blackwell, A.F., K.N. Whitley, J. Good, and M. Petre, "Cognitive Factors in Programming with Diagrams", accepted for publication in *Artificial Intelligence Review*.
- [2] Card K., D. MacKinlay and B. Shneiderman (Eds), *Readings in Information visualization, Using Vision to Think*, Morgan Kaufmann, 2000.
- [3] Casner, S. and J.H. Larkin, "Cognitive Efficiency Considerations for Good Graphic Design", *11th Annual Conference of the Cognitive Science Society*, August 1989.
- [4] Casner, S., "A Task-Analytic Approach to the Automated Design of Graphic Presentations". *ACM Trans. on Graphics*, 10:111-151, April 1991.
- [5] Clarke, E. and J. Wing, "Formal Methods: State of the Art and Future Directions", *ACM Computing Surveys*, vol 28, no. 4, December 1996, pp. 626-43.
- [6] Conklin, J., "Hypertext: An introduction and survey", *IEEE Comput.* 20,9 (Sept. 1987), 17-40
- [7] Dulac, Nicolas, Thomas Viguier, Nancy Leveson, and Margaret-Anne Storey. "On the Use of Visualization in Formal Requirements Specification." *Proceedings of the International Conference on Requirements Engineering*. Essen, Germany. Sep 2002.
- [8] Fitter, M. and T.R.G. Green, "When do diagrams make good computer languages", *International Journal of Man-Machine Studies*, 11(2):235-261, March 1979.
- [9] Gerhart, S.L., D. Craigen, and T. Ralston. "Observations on industrial practice using formal methods", *Proceedings 15th International Conference on Software Engineering (ICSE)*, Baltimore, Maryland, USA, May 1993.
- [10] Green, T.R.G., "Conditional Programs Statements and their Comprehensibility to Professional Programmers", *Journal of Occupational Psychology*, 50, 93-109, 1977
- [11] Green, T.R.G. and A.F. Blackwell, "A tutorial on cognitive dimensions". (1998)
- [12] Guttag, John V., James J. Horning, S.J. Garland, K.D. Jones, A. Modet, and J.M. Wing. "Larch: Languages and Tools for Formal Specification." Springer-Verlag, New York, NY, 1993.
- [13] Harel, David. "Statecharts: A visual Formalism for Complex Systems", *Science of Computer Programming* 8. Elsevier Science Publishers B.V., Nirth Holland. 1987. pp. 231-274.
- [14] Heimdahl, Mats P.E. and Nancy Leveson. "Completeness and Consistency Analysis of State-Based Requirements." *IEEE Transactions on Software Engineering*. May 1996.
- [15] Heimdahl, Mats P.E. and Michael W. Whalen. "Reduction and Slicing of Hierarchical State Machines", *Proceedings of the Sixth European Software Engineering Conference (ESEC/FSE 97)*.

- [16] Hutchins, Edwin, James Hollan, and Donald Norman, “Direct Manipulation Interfaces”. In Donald Norman and Stephen Draper, *User Centered System Design*, 1986, pp. 87-124
- [17] Johnson-Laird, P.N. “Mental Models”, In M.I. Posner, Ed., *Foundations of Cognitive Science*. MIT Press, Cambridge, MA, 1989, 468-499.
- [18] Larkin, J.H. and H.A. Simon, “Why a Diagram is (Sometimes) Worth Ten Thousand Word”, *Cognitive Science*, 11(1):65-99, Jan-Mar 1987.
- [19] Leveson, Nancy, Mats P.E. Heimdahl, Holly Hildreth, and Jon D. Reese. “Requirements Specification for Process-Control Systems”, Published in *IEEE Transactions on Software Engineering*, September 1994.
- [20] Leveson, Nancy. “Safeware: System Safety and Computers”. Addison-Wesley Publishing Company, 1995.
- [21] Leveson, Nancy, Jon Damon Reese and Mats P.E. Heimdahl. “SpecTRM: A CAD System for Digital Automation”. Safeware Engineering Corporation, 1998.
- [22] Leveson, Nancy. “Intent Specifications: An Approach to Building Human-Centered Specifications.” *IEEE Transactions on Software Engineering*. Jan 2000.
- [23] Leveson, Nancy. “Completeness in Formal Specification Language Design for Process-Control Systems.” *Proceedings of Formal Methods in Software Practice Conference*. Portland, OR. Aug 2000.
- [24] Petre, M. “Why Looking Isn't Always Seeing: Readership Skills and Graphical Programming.” *Communications of the ACM*, 38(6), pp.33-44, 1995.
- [25] Nelson, T.H. “Xanalogical Structure, Needed Now More than Ever: Parallel Documents, Deep Links to Content, Deep Versioning and Deep Re-Use”, Project Xanadu and Keio University.
- [26] Ranson, D.S. and D.D. Woods, “Making Automation Activity Visible”, Technical Report 1995-03, Ohio State University, December 1995.
- [27] Rasmussen, J. and A. Jensen. “A study of mental procedures in electronic troubleshooting”. Roskilde, Denmark: Danish Atomic Energy Commission, Research Establishment Riso, 1973.
- [28] Rasmussen, J. and A. Jensen. “Mental procedures in real-life tasks: A case study of electronic troubleshooting”. *Ergonomics*, 17, 293-307, 1974.
- [29] Rasmussen, J. “Information Processing and Human-Machine Interaction : An Approach to Cognitive Engineering.” North Holland, 1986.
- [30] Safeware Engineering Corporation. *SpecTRM User Manual*. Version 1.0.0. 2003.
- [31] Shore, John. “The Sachertorte Algorithm and Other Antidotes to Computer Anxiety”. Penguin Books, New York, 1986.

- [32] Spivey, J.M. "The Z Notation: A Reference Manual." Prentice-Hall, Englewood Cliffs, New Jersey, USA, Second edition, 1992.
- [33] Storey, M.A., and H.A. Muller, "Manipulating and documenting software structures using ShriMP Views". In *Proceedings of the 1995 International Conference on Software Maintenance*. (ICSM '95) Opio (Nice), France October 16-20, 1995.
- [34] Storey, M.A., F.D. Fracchia, and H.A. Muller, "Cognitive Design Elements to Support the Construction of a Mental Model during Software Exploration", *Journal of Software Systems*, 44:171-185, 1999.
- [35] Thüring, M., J. Hannemann and J. Haake, "Hypermedia and Cognition: Designing for Comprehension", *Communications of the ACM* 38(8), 1995, 57-69.
- [36] Tufte, E.R. "The Visual Display of Quantitative Information". Graphics Press. 1983.
- [37] Vicente, Kim J., "Cognitive Work Analysis: Toward Safe, Productive, and Healthy Computer-Based Work". Lawrence Erlbaum Associates, Publishers. Mahwah, New Jersey, 1999.
- [38] Viguiet, Thomas. "Evaluating Visualization In Formal Requirements Specification: An Experiment With Human Subjects". M.S. Thesis, Massachusetts Institute of Technology, 2003.
- [39] Woods, David. "Toward a theoretical base for representation design in the computer medium: Ecological perception and aiding human cognition." In J.M. Flach et al., editors *An ecological Approach to Human Machine Systems I: A Global Perspective*, Erlbaum, Hillsdale, New Jersey, 1995.
- [40] Zimmerman, Marc. "Investigating the Readability of Formal Specification Languages". M.S. Thesis, Massachusetts Institute of Technology, 2001.
- [41] Zimmerman, Marc, Kristina Lundqvist and Nancy Leveson, "Investigating the Readability of State-Based Formal Requirements Specification Languages", *International Conference on Software Engineering*, Orlando, May 2002.

Appendix A

Full-Page Figures

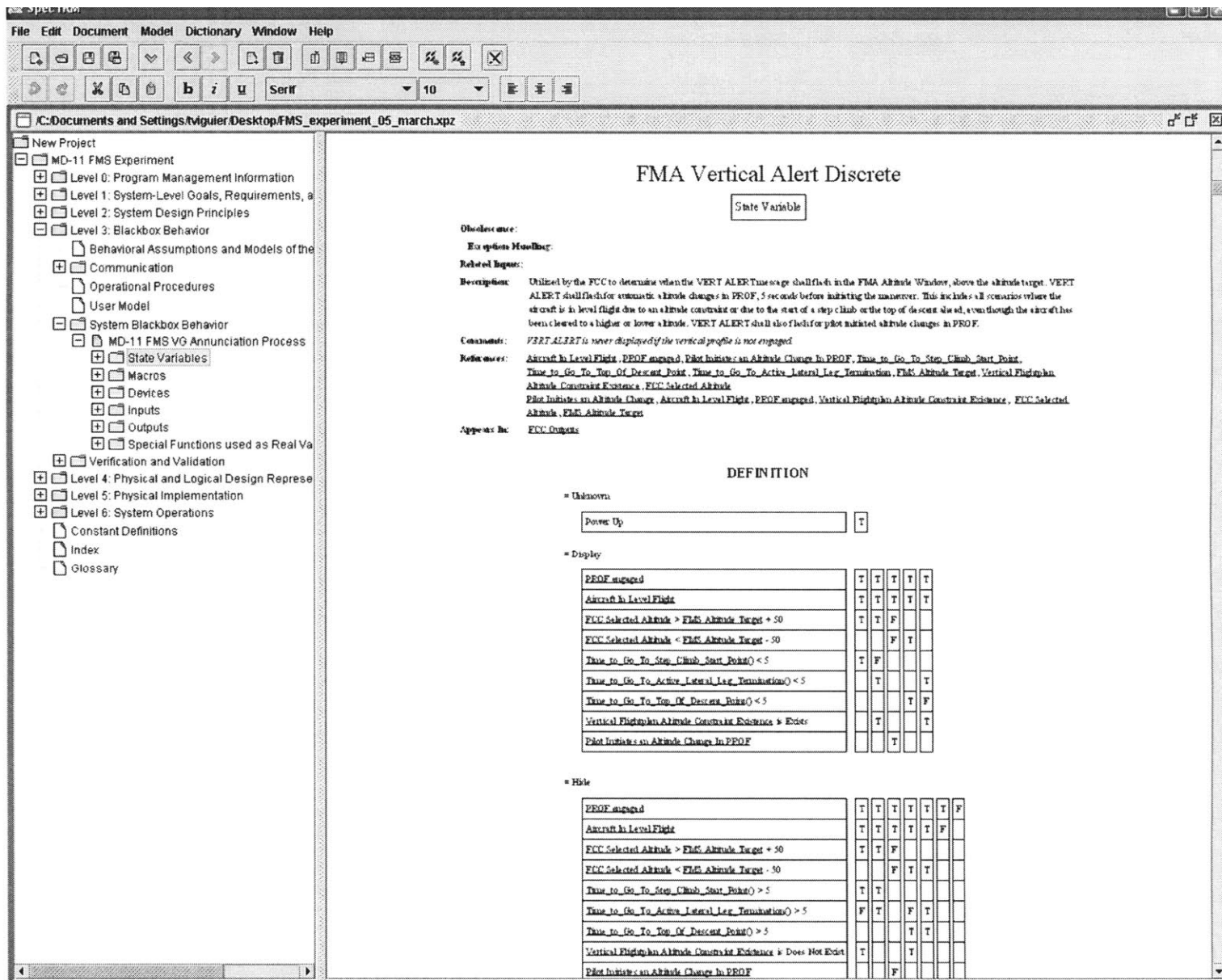


Figure A: Screen capture of the SpecTRM GUI.

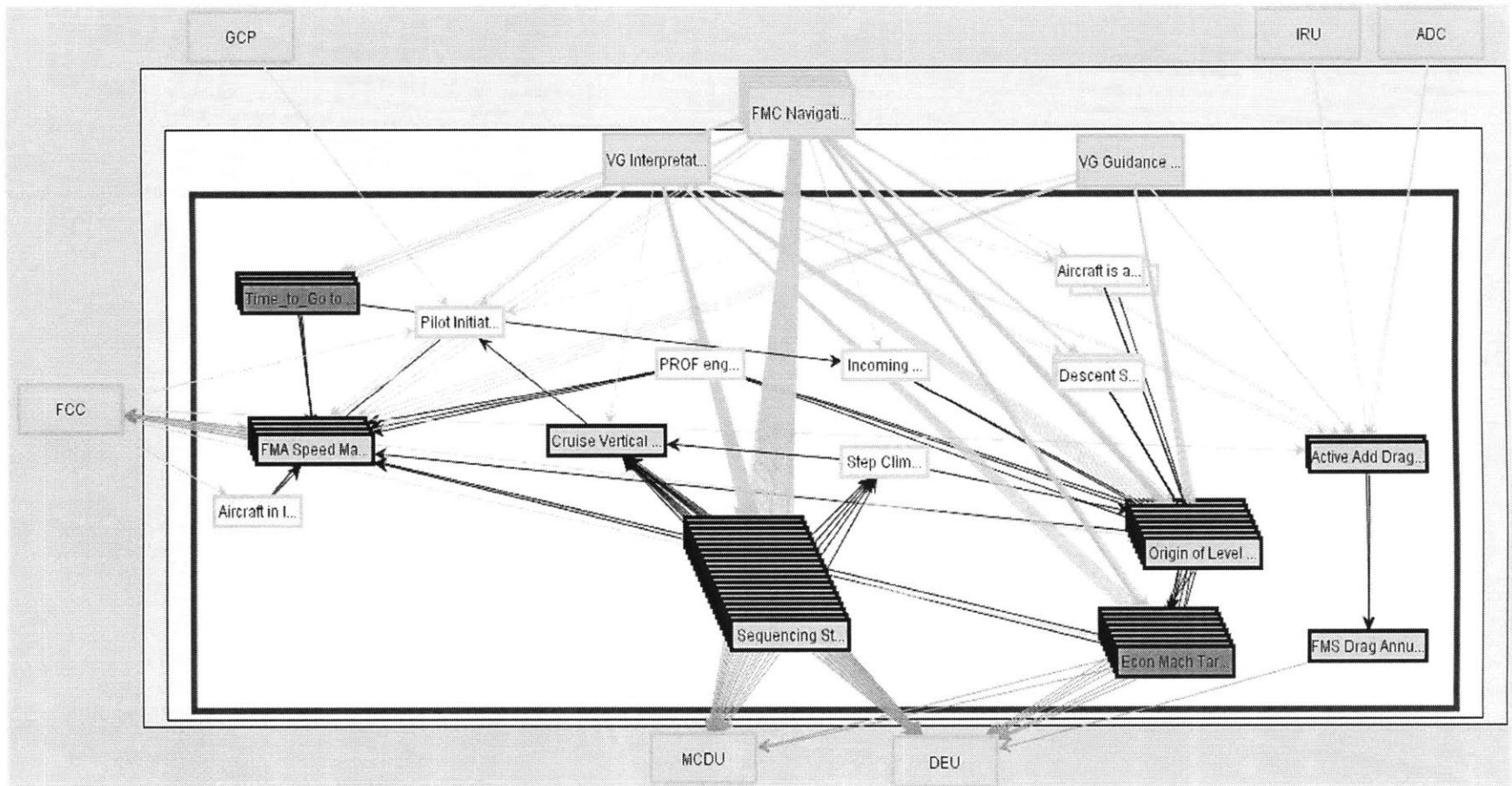


Figure B: An organized overview of the view of the structure of the specification (V1).

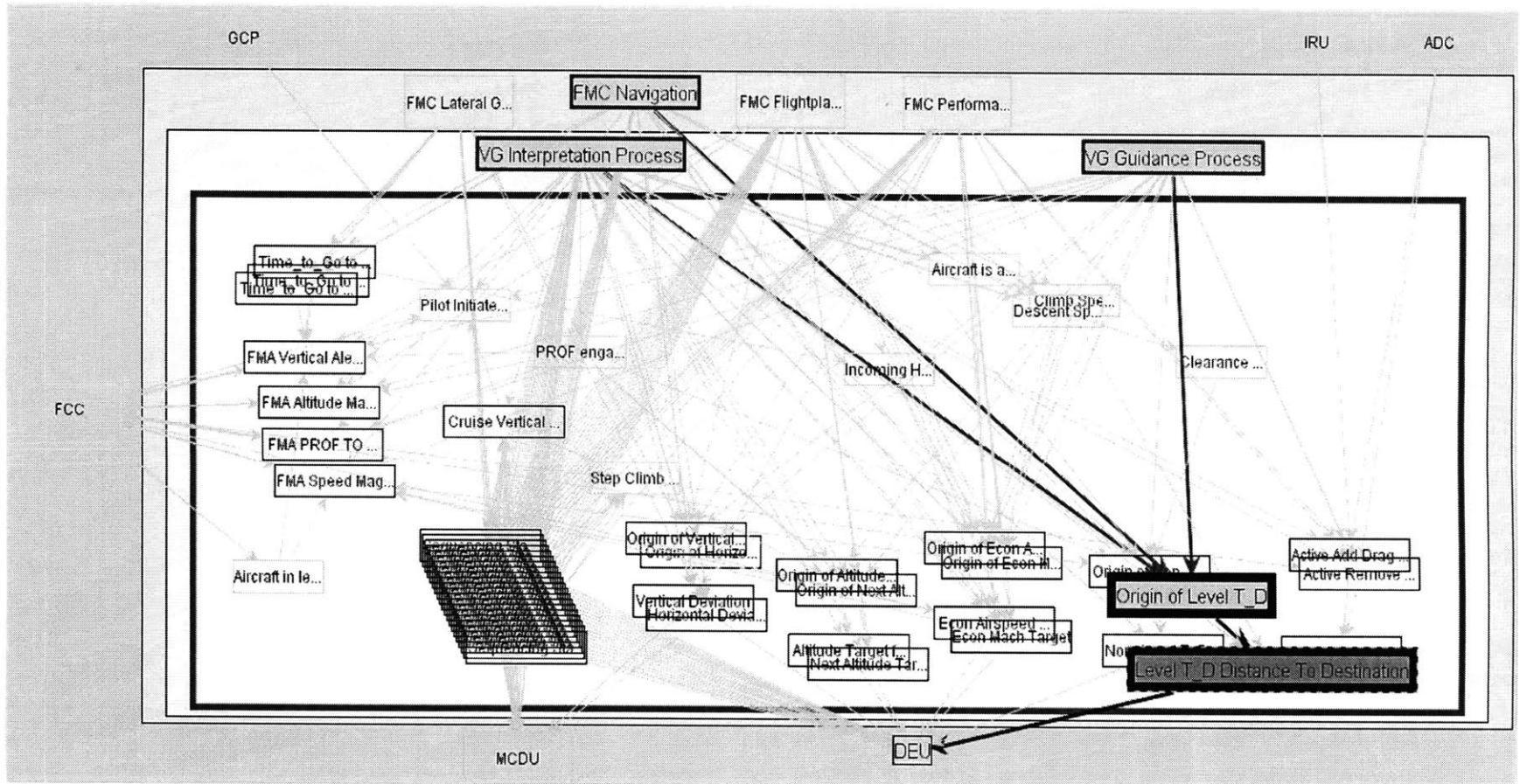


Figure C: Sliced structural overview. The state variable *Origin of Level T_D* and its structural dependencies are emphasized over the rest of the model, which is preserved as context.

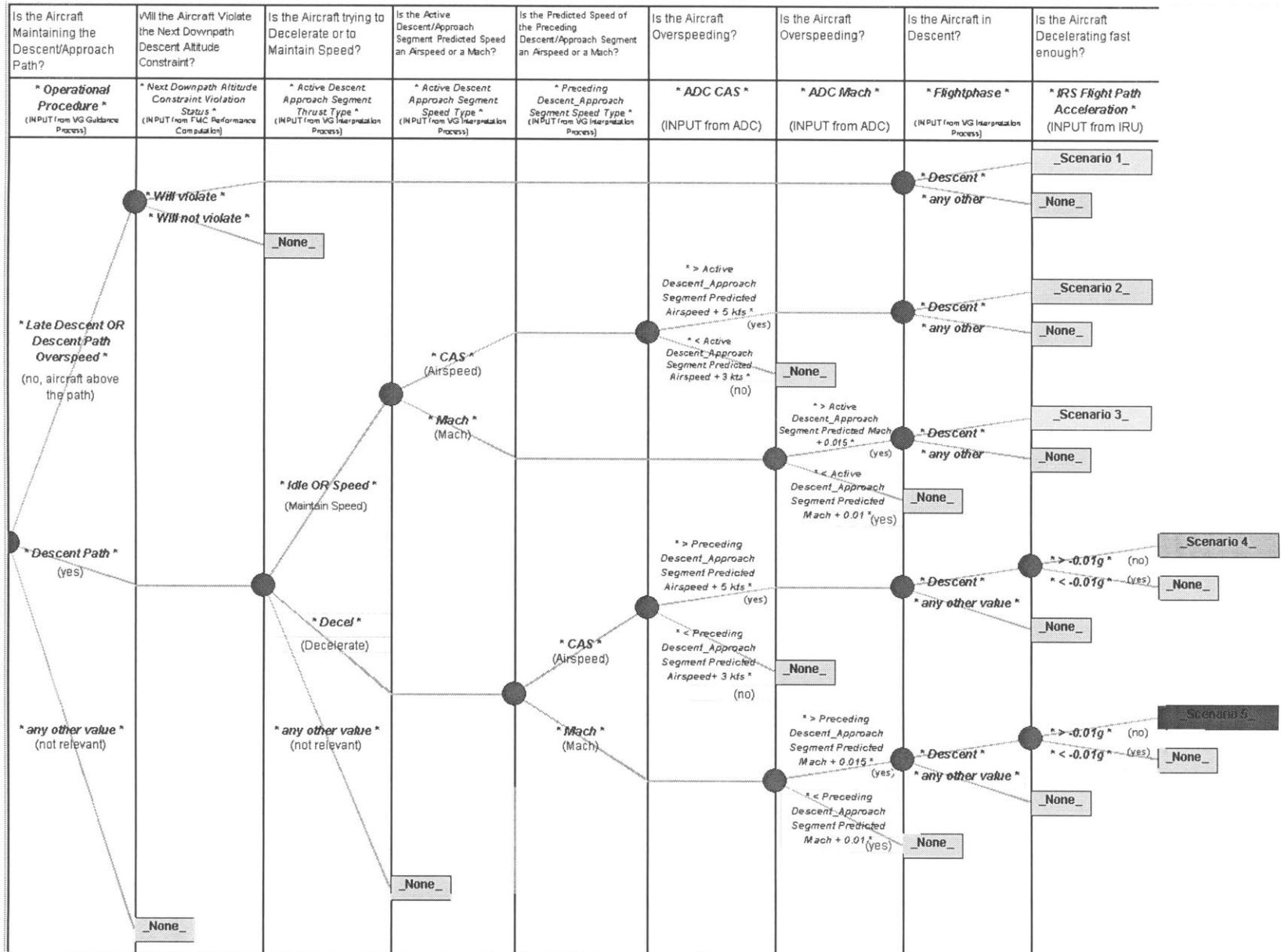


Figure D: A Questions-based Decision Tree taken from the MD-11 Vertical Guidance Specification. The state variable *Active Add Drag Scenario* indicates which of the five possible scenarios for extending airbrakes is active, if any.

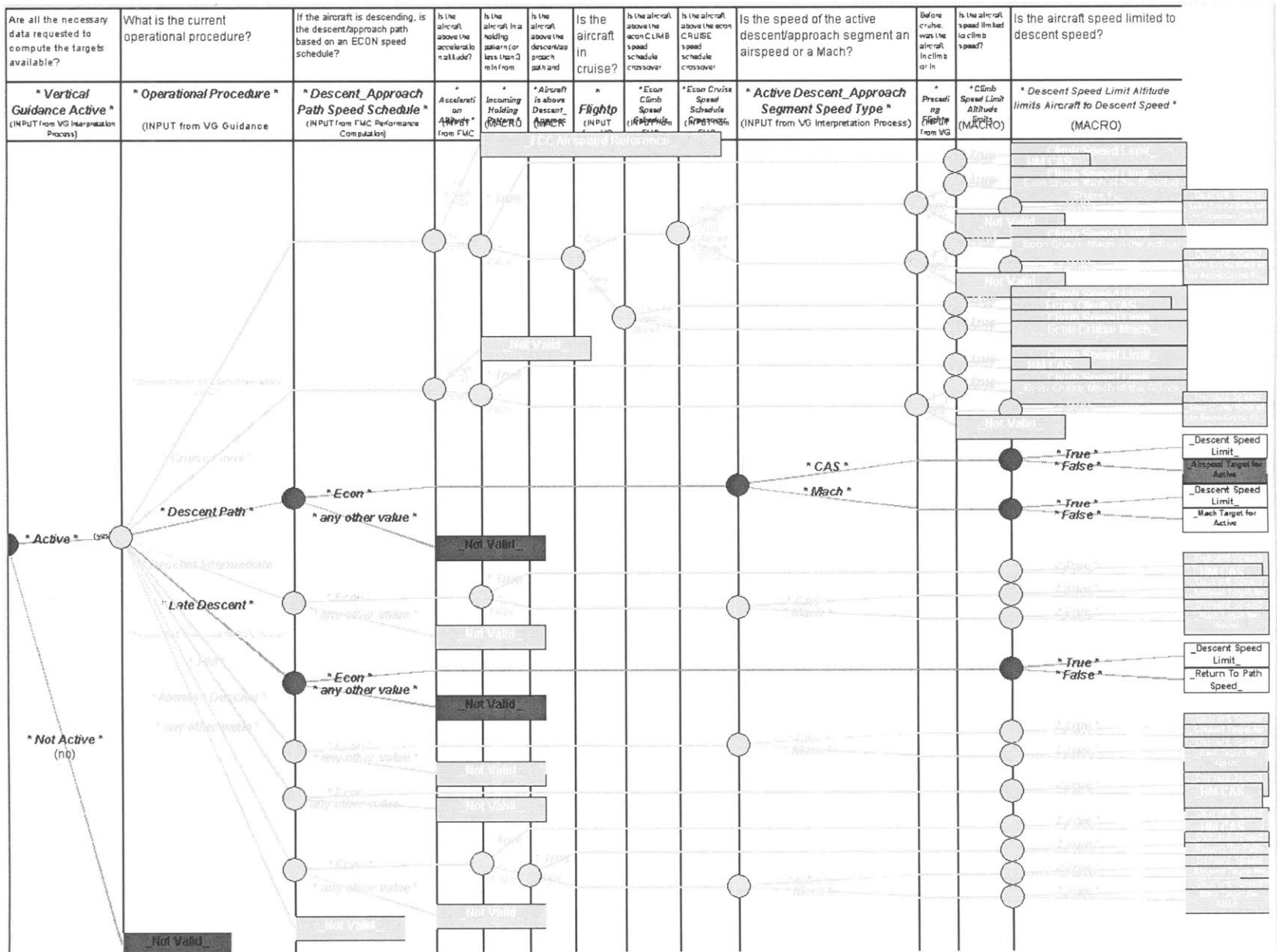


Figure F: Effects of a slicing scenario applied on the state variable *Origin of Econ Airspeed Target* (Figure E). The states reachable under this scenario are highlighted. The active scenario specifies that the flightphase is 'Descent' and that the current operational procedure is 'Descent Path' or 'Late Descent'.

= Scenario_1

Operational Procedure is Late Descent	T	
Operational Procedure is Descent Path Overspeed		T
Next Downpath Altitude Constraint Violation Status is Will Not Violate	T	T
Airbrakes Extension Status is Extended	T	T

= Scenario_2

Operational Procedure is Descent Path	T	T	T
Active Descent Approach Segment Thrust Type is Idle	T		
Active Descent Approach Segment Thrust Type is Decel		T	
Active Descent Approach Segment Thrust Type is Speed			T
Active Descent Approach Segment Speed Type is CAS	T	T	T
ADC CAS < Active Descent Approach Segment Predicted Airspeed - 5	T	T	T
Airbrakes Extension Status is Extended	T	T	T

= Scenario_3

Operational Procedure is Descent Path	T	T
Active Descent Approach Segment Thrust Type is Idle	T	
Active Descent Approach Segment Thrust Type is Decel		T
Active Descent Approach Segment Speed Type is Mach	T	T
ADC Mach < Active Descent Approach Segment Predicted Mach - 0.015	T	T
Airbrakes Extension Status is Extended	T	T

= Scenario_4

Operational Procedure is Descent Path	T
Active Descent Approach Segment Thrust Type is Speed	T
Active Descent Approach Segment Speed Type is CAS	T
ADC CAS < Active Descent Approach Segment Predicted Airspeed - 3	T
Airbrakes Extension Status is Extended	T

= Scenario_5

Operational Procedure is Descent Path	T
Active Descent Approach Segment Thrust Type is Speed	T
Active Descent Approach Segment Speed Type is Mach	T
ADC Mach < Active Descent Approach Segment Predicted Mach - 0.007	T
Airbrakes Extension Status is Extended	T

Figure G: Five of the six AND/OR tables necessary to describe the state variable “Active Add Drag Scenario” in the specification of the MD-11 Vertical Guidance Annunciation Process. This state variable specifies which of the five possible scenarios for extending airbrakes is active, if any.

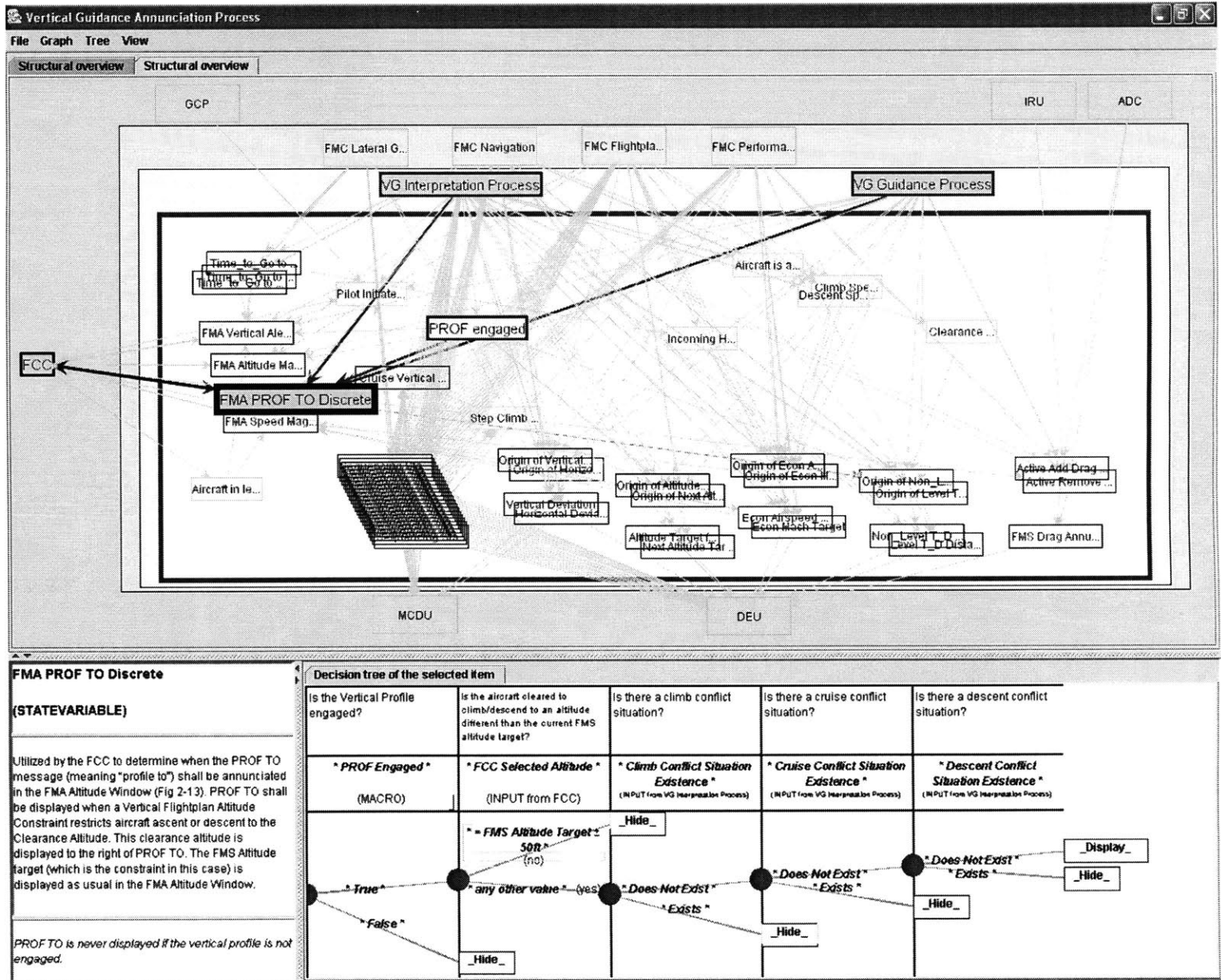


Figure H: Screen capture of the entire visualization tool.

Appendix B

Experiment Questions

Tutorial Question

State Values

- Sequencing Status of Step 2 End in primary flightplan = “Does not Exist”

Macro

- PROF Engaged = True

Inputs from VG Interpretation Process

- Flightphase = Climb
- Clearance Altitude = 11000 ft

Input from VG Guidance Process

- Operational Procedure = Cruise Level

Input from FMC Flightplanning

- Step_1_End_Altitude = 12500 ft
- Active Cruise Flightlevel = 12000 ft

Info Box Z

2- Given the information in “Info Box Z”

a) What additional information is necessary for the State Variable **Origin of Next Altitude Target For Display on PFD** to transition to the State “Active Cruise Flightlevel”?

Element	Element Type	Value
Step climb exists in primary Vertical Flightplan	Macro	True
Vertical Guidance Active	Input	Active

b) Given all the above information including what you found in a), if you can infer the state of the following State Variables, write it in the space below, otherwise, leave it blank.

State Variable	State
Sequencing Status of Step 1 End in Primary Flightplan	Unsequenced
Sequencing Status of Step 2 End in Primary Flightplan	Does not Exist
Sequencing Status of Step 3 End in Primary Flightplan	
Sequencing Status of Step 4 End in Primary Flightplan	
Sequencing Status of Step 5 End in Primary Flightplan	
Origin Of Altitude Target For Display On PFD	VG Reference Altitude

c) If ALL the above information is exact except for the **Flightphase**, which just changed to **Cruise**, what do we know about the **Aircraft Altitude**?

Aircraft Altitude < 12400 ft

Appendix C

Complete Quantitative Results

		Q1		Q2		Q3	
		Part 1	Part 2	Part 1	Part 2	Part 1	Part 2
Subject #1	Tool	S		V		S+V	
EECS	Result	3	8	5	10	6	10
	Difficulty Rating	9	8	5	3	3	1
	Time (min)	21	21	21	7	21	7
Subject #2	Tool	S+V		S		V	
Aerospace	Result	6	8	2	6	5	4
	Difficulty Rating	7	3	10	5	5	4
	Time (min)	18	15	22	7	17	8
Subject #3	Tool	V		S+V		S	
Aerospace	Result	8	5	3	8	6	10
	Difficulty Rating	8	9	8	5	8	2
	Time (min)	23	21	21	11	24	9
Subject #4	Tool	S		S+V		V	
Aerospace	Result	8	6	8	8	7	10
	Difficulty Rating	8	9	6	5	6	5
	Time (min)	21	21	21	20	17	18
Subject #5	Tool	S+V		V		S	
EECS	Result	8	10	7	7	1	1
	Difficulty Rating	5	9	5	2	5	2
	Time (min)	17	21	18	10	12	5
Subject #6	Tool	V		S		S+V	
EECS	Result	8	3	9	4	7	10
	Difficulty Rating	5	7	5	8	5	6
	Time (min)	15	15	14	15	12	7
Subject #7	Tool	S		V		S+V	
EECS	Result	10	10	10	10	10	10
	Difficulty Rating	2	2	3	3	7	2
	Time (min)	14	15	11	3	13	7
Subject #8	Tool	S+V		S		V	
Aerospace	Result	10	9	0	8	7	10
	Difficulty Rating	10	5	10	5	10	1
	Time (min)	20	17	16	17	21	10
Subject #9	Tool	V		S+V		S	
EECS	Result	4	3	0	7	4	0
	Difficulty Rating	10	9	10	5	6	X
	Time (min)	26	31	15	12	20	?
Subject #10	Tool	S		S+V		V	
EECS	Result	4	7	4	4	7	8
	Difficulty Rating	5	8	3	2	4	7
	Time (min)	12	19	10	4	15	11
Subject #11	Tool	S+V		V		S	
Aerospace	Result	7	10	3	4	8	6
	Difficulty Rating	6	4	8	2	5	3
	Time (min)	22	22	22	3	11	9
Subject #12	Tool	V		S		S+V	
Aerospace	Result	5	9	8	8	6	6
	Difficulty Rating	7	8	6	5	8	6
	Time (min)	12	16	13	11	16	9

